UNIVERSIDAD POLITÉCNICA DE MADRID
ESCUELA TÉCNICA SUPERIOR DE INGENIEROS EN TOPOGRAFÍA, GEODESIA Y CARTOGRAFÍA
MÁSTER EN INGENIERÍA GEODÉSICA Y CARTOGRAFÍA

TRABAJO FIN DE MÁSTER

ESTUDIO DE LAS POSIBILIDADES DE LOS DATOS ABIERTOS ENLAZADOS (LINKED OPEN DATA) PARA LA REALIZACIÓN DE MASHUPS DE ÁMBITO GEOGRÁFICO.

Madrid, Julio de 2014

Alumno: Izar Sinde González

Tutores: Miguel Ángel Manso Callejo
Ramón P. Alcarria Garrido

A mi abuela.

Estés donde estés, eres parte de esto.

Agradecimientos

Quisiera agradecer primeramente al tutor de este proyecto, Ramón Alcarria, toda la ayuda prestada, que no fue poca, así como todos los ánimos que me ha transmitido a lo largo de la realización de este estudio. Al mismo tiempo, como cotutor Miguel Ángel Manso, quisiera agradecerle la oportunidad de realizar este estudio a la vez que toda su ayuda.

En segundo lugar agradecer a Javier Lacasta su ayuda resolviendo algunos de los problemas que fueron surgiendo a lo largo del trabajo, siempre de forma diligente.

A mis padres, a los que debo todo, quisiera agradecerles el apoyo y la confianza que han depositado en mí a lo largo de los últimos años.

Rosana, siempre a mi lado en este tiempo. Ayudándome y soportándome en los malos momentos. Disfrutando y riendo en los buenos. Gracias.

Finalmente me gustaría agradecer a mis compañeros y nuevos amigos toda la alegría, buen ánimo y apoyo que me han transmitido durante la realización de este Máster.

Resumen

La web ha sufrido una drástica transformación en los últimos años, debido principalmente a su popularización y a la enorme cantidad de información que alberga. Debido a estos factores se ha dado el salto de la denominada Web de Documentos, a la Web Semántica, donde toda la información está relacionada con otra. Las principales ventajas de la información enlazada estriban en la facilidad de reutilización, accesibilidad y disponibilidad para ser encontrada por el usuario.

En este trabajo se pretende poner de manifiesto la utilidad de los datos enlazados aplicados al ámbito geográfico y mostrar como pueden ser empleados hoy en día. Para ello se han explotado datos enlazados de carácter espacial provenientes de diferentes fuentes, a través de servidores externos o *endpoints* SPARQL. Además de eso se ha trabajado con un servidor privado capaz de proporcionar información enlazada almacenada en un equipo personal.

La explotación de información enlazada se ha implementado en una aplicación web en lenguaje JavaScript, tratando de abstraer totalmente al usuario del tratamiento de los datos a nivel interno de la aplicación. Esta aplicación cuenta además con algunos módulos y opciones capaces de interactuar con las consultas realizadas a los servidores, consiguiendo un entorno más intuitivo y agradable para el usuario.

Abstract

In recent years the web has suffered a drastic transformation because of the popularization and the huge amount of stored information. Due to these factors it has gone from Documents web to Semantic web, where the data are linked. The main advantages of Linked Data lie in the ease of his reuse, accessibility and availability to be located by users.

The aim of this research is to highlight the usefulness of the geographic linked data and show how can be used at present time. To get this, the spatial linked data coming from several sources have been managed through external servers or also called endpoints. Besides, it has been worked with a private server able to provide linked data stored in a personal computer.

The use of linked data has been implemented in a JavaScript web application, trying completely to abstract the internally data treatment of the application to make the user ignore it. This application has some modules and options that are able to interact with the queries made to the servers, getting a more intuitive and kind environment for users.

Índice

Índice de figuras

Índice de tablas

1. Introducción

En la actualidad existen ingentes cantidades de información de todo tipo disponible a través de la web [1] y en lo concerniente a la información geográfica no es diferente. Toda esta información se encuentra almacenada en repositorios de información de forma muy heterogénea y sin ningún tipo de nexo de unión, lo cual dificulta la búsqueda, el acceso y en definitiva el uso de los mismo de la forma adecuada.

Además de esto, mucha de la información que se encuentra en la web aparece representada en formatos que, lejos de estar disponibles para todo el mundo de forma gratuita, están sujetos a una serie de restricciones que del mismo modo que el inconveniente anterior, dificulta el uso y explotación de la información [2].

Para soluciona los problemas anteriormente citados, ya en 1989 Tim BernesLee formuló el concepto de *Open Liked Data* o Información abierta y enlazada. Esta consistía básicamente en una estructura de conocimientos interconectado que enlazaba la información y por tanto aunaba el concepto de datos abiertos (a todo usuario que desee emplearlos) con el hecho de que distintas fuentes de información tengan sus conceptos enlazados y por tanto fácilmente accesibles. De este modo se consigue aumentar el valor de los datos al reutilizar la información y facilita el hallazgo de los datos que realmente se buscan.

Basado en estos conceptos y en el contexto de la titulación Master en Ingeniería Geodésica y Cartografía surge el presente Trabajo Fin de Máster (TFM).

Este Trabajo Fin de Máster pretende poner de manifiesto la utilidad de la tecnología Open Linked Data para demostrar la capacidad de solucionar los problemas mencionados en el ámbito geográfico.

Esto se logrará desarrollando una aplicación orientada a la web, en lenguaje *JavaScript*. Esta aplicación tratará de explotar una serie de datos geográficos seleccionados dentro de algunas de las iniciativas de *Open Linked Data* más reseñables el ámbito español, como son las del *Ontology Engeneering Group* (datos de AEMET y del IGN) o la del Ayuntamiento de Gijón, al igual que algunas en el ámbito internacional como DBpedia.

De este modo se conseguirá profundizar en los conocimientos tratados en las asignaturas de Aplicaciones distribuidas para la información geográfica y Programación web del máster.

2. Objetivos

El objetivo principal de este estudio es **ilustrar el manejo de datos abierto enlazados (Open Linked Data) en formato Resource Description Framework (RDF) y poner de manifiesto las posibilidades y el valor añadido que la semántica proporciona a los datos,** tanto geográficos como de cualquier tipo.

Este objetivo principal se puede traducirlo en los siguientes objetivos operativos:

1. Obtención de información geográfica enlazada a partir de servidores externos (*endpoint*) con ayuda de peticiones con el protocolo Hypertext Transfer Protocol (HTTP) sin ayuda de una plataforma específica de tratamiento de datos RDF.
2. Obtención de información geográfica enlazada de servidores externos (endpoint) a través de un servidor propio en el que se instale un *software* especializado en la explotación y visualización de datos enlazados.
3. Obtención de información geográfica enlazada a partir de un repositorio personal en el que se almacene información espacial enlazada propia. Esto se conseguirá con la ayuda de un servidor propio sobre el que se instalará el *software* especializado, que además de consultar a servidores externos, permite consultar datos propios almacenados en el propio servidor.
4. Una vez obtenida la información geográfica enlazada de distintas fuentes y con diferentes temáticas de diferentes documentos RDF o servidores externos, se tratará de realizar su representación en un visualizador geográfico. Para ello se explotará la tecnología de Openlayers y el servicio de mapas de Openlayers.
5. Implementación de una serie de menús interactivos que interactúen con las consultas y permitan al usuario un manejo más intuitivo de la información espacial enlazada.
6. Implementación de un módulo que permita fijar varios contextos geográficos de trabajo, y así focalizar los objetivos de las consultas, lo que aumentará la usabilidad de la aplicación.

3. Antecedentes

El contenido de este apartado versará primeramente sobre la evolución de la web y la aparición de la web semántica. Después se explicarán en profundidad los conceptos de datos abiertos y enlazados (Open Linked Data) y se enumerarán las principales iniciativas existentes al respecto, existentes. Finalmente se hará un análisis de las principales plataformas para la explotación de datos enlazados

3.1. Web de documentos

La web de documentos se corresponde a la segunda generación web basada en comunidades de usuarios. Se pasó de una web informativa, creada por expertos, a una web social y colaborativa, donde cualquiera puede participar fácilmente. Aparecen así mismo aplicaciones web muy potentes y sencillas de manejar con gran usabilidad (enfocadas al usuario final).

Su desarrollo se basa en los Content Management System (CMS) o Sistemas de Gestión de Contenidos, que permiten la creación y administración de contenidos principalmente en páginas web. Primero surgieron las páginas estáticas (HTML) y luego las páginas dinámicas (CGI, PHP, ASP, Java).

Las principales características de la web 2.0 son, por tanto, el protagonismo del usuario, la participación del mismo y la usabilidad de sus herramientas.

Algunas de las tecnologías características de esta web de documentos son el HTML para presentar datos, el Document Object Model (DOM) para mostrar e interactuar dinámicamente con la información, el lenguaje XML para intercambiar y manipular datos, el XMLHttpRequest para recuperar y enviar datos de modo asíncrono y el lenguaje de programación JavaScript como nexo de unión.

Así mismo se crean los blogs, wikis y el Really Simple Syndication (RSS).

3.2. Web semántica

Es la web que disponible actualmente. Está compuesta principalmente de documentos HTML en lenguaje natural y multimedia. De aquí surge la novedad de esta nueva web, en la que se puede encontrar una información determinada o integrarla.

Antes de la web semántica, surge la web sintáctica, en la que aparecen un conjunto de recursos enlazados entre sí (páginas HTML enlazadas por referencias). Se caracteriza porque no se enlazan todas las páginas existentes, la escasa precisión de los resultados y la alta sensibilidad al vocabulario empleado en la búsqueda.

De esta filosofía nace la web semántica, que añade la semántica que le falta a la web sintáctica para crear un entorno donde se puede acceder a la información que se necesita de

un modo exacto y completo, facilitando de este modo el procesado de la información y la resolución de problemas de interoperabilidad entre aplicaciones.

Una definición adecuada para esta web es la aportada por el Consorcio de la Web (W3C) que dice:

"La Web Semántica es una Web extendida, dotada de mayor significado en la que cualquier usuario en Internet podrá encontrar respuestas a sus preguntas de forma más rápida y sencilla gracias a una información mejor definida. Al dotar a la Web de más significado y, por lo tanto, de más semántica, se pueden obtener soluciones a problemas habituales en la búsqueda de información gracias a la utilización de una infraestructura común, mediante la cual, es posible compartir, procesar y transferir información de forma sencilla. Esta Web extendida y basada en el significado, se apoya en lenguajes universales que resuelven los problemas ocasionados por una Web carente de semántica en la que, en ocasiones, el acceso a la información se convierte en una tarea difícil y frustrante." [3]

El objetivo, por tanto, de la web semántica es crear un medio universal que permita el intercambio de datos y brindar un mayor significado a los mismos para que puedan ser interpretados por las máquinas [4].

3.3. Linked data

Linked data es un término utilizado para describir las mejores prácticas recomendadas para exponer, compartir e integrar conjuntos de datos en la Web Semántica. Los cuatro principios fundamentales a cumplir por el Linked data son el de usar URIs para identificar elementos o conceptos, URIs HTTP, ofrecer información sobre los recursos con el lenguaje RDF e incluir enlaces a otros elementos (URI). Un Uniform Resource Identifier o identificador de recursos uniforme (URI) es una cadena de caracteres cuyos componentes son el protocolo de acceso al recurso (http), la autoridad de nombres y la ruta y que identifica los recursos de una red de forma unívoca.

Ejemplo e URI: http://es.dbpedia.org/page/Madrid

Protocolo de acceso: http://

Autoridad de nombre: es.dbpedia.org

Ruta: page/Madrid

El lenguaje RDF es una familia de especificaciones de la W3C, que fue ideado como modelo de datos para metadatos pero que en la actualidad también se utiliza como método general para la descripción de conceptos o recursos web.

Linked data es en general la exposición de datos en formato RDF y enlazados con otros datos.

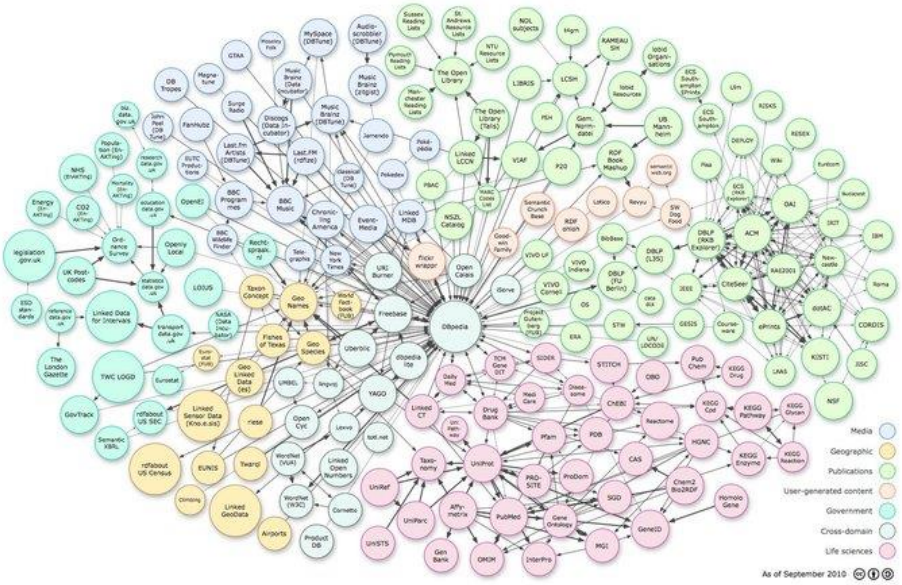


Figura 1. Esquema *Linked Data*

El proceso que se sigue para transformar la web de documentos a la web semántica sigue diferentes etapas, comenzando con la especificación de los conceptos, el modelado de los mismos, la generación de documentos RDF, la generación de los enlaces, la publicación y, finalmente, explotación de los datos. En el presente estudio se va a trabajar la parte de la explotación de los datos semánticos.

Para la utilización de los datos semánticos existen diversos lenguajes de consulta de datos enlazados. El más extendido es el lenguaje SPARQL (SPARQL Protocol and RDF Query Languaje), que además de estar estandarizado para consultas a RDF, está normalizado por el RDF Data Access Working Group (DAWG) del W3C. Guarda cierta similitud con el lenguaje de consulta a bases de datos Structured Query Language (SQL), pero se diferencia en que las sentencias que utiliza para la consulta se basan en tripletas de conceptos (Sujeto, Predicado, Objeto), como se explicará más adelante. El lenguaje SPARQL asume que no se puede garantizar la completitud de la información descubierta, es decir, si algo no se sabe, es desconocido, no falso. Además de eso permite consultar a múltiples colecciones para posibilitar la integración de la información. En la actualidad existe la versión SPARQL 1.0 y en borrador la SPARQL 1.1, la cual no está implementada en todos los servidores de datos RDF.

El uso del lenguaje SPARQL puede llevarse a cabo de dos modos, uno de ellos a través de consultas vía Application Programming Interface (API) de repositorios RDF (Jena2, Sesame, OpenLinkVirtuoso…) o a través de consulta web a un punto de entrada (*endpoint*) SPARQL. En el presente estudio se emplearán los dos modos de consulta.

La aplicación del mundo geográfico al *linked data* (Figura 2) conlleva la adhesión de información espacial como propiedades RDF, lo cual provoca la definición de un esquema RDF y una extensión de los lenguajes de consulta para ellos. Existen diversos proyectos para lograr esto, como stSPARQL O W3C GEO *vocabulary*, pero el más extendido es GEOSPARQL. Este estándar para la representación y consulta de *linked data* espacial fue propuesto por el Open Geospatial Consortium (OGC). Indica un conjunto de propiedades y relaciones espaciales para

tratar datos en formato de representación de geometrías GML o WKT. Además de esto supone una extensión del lenguaje SPARQL para consulta de datos geográficos. Está basado en las siguientes normas:

- ISO 19109: Modelo general de features
- ISO 19125: Simple Features
- ISO 19107: Features geometry
- ISO 13249: Lenguajes de bases de datos- SQL Multimedia parte espacial.

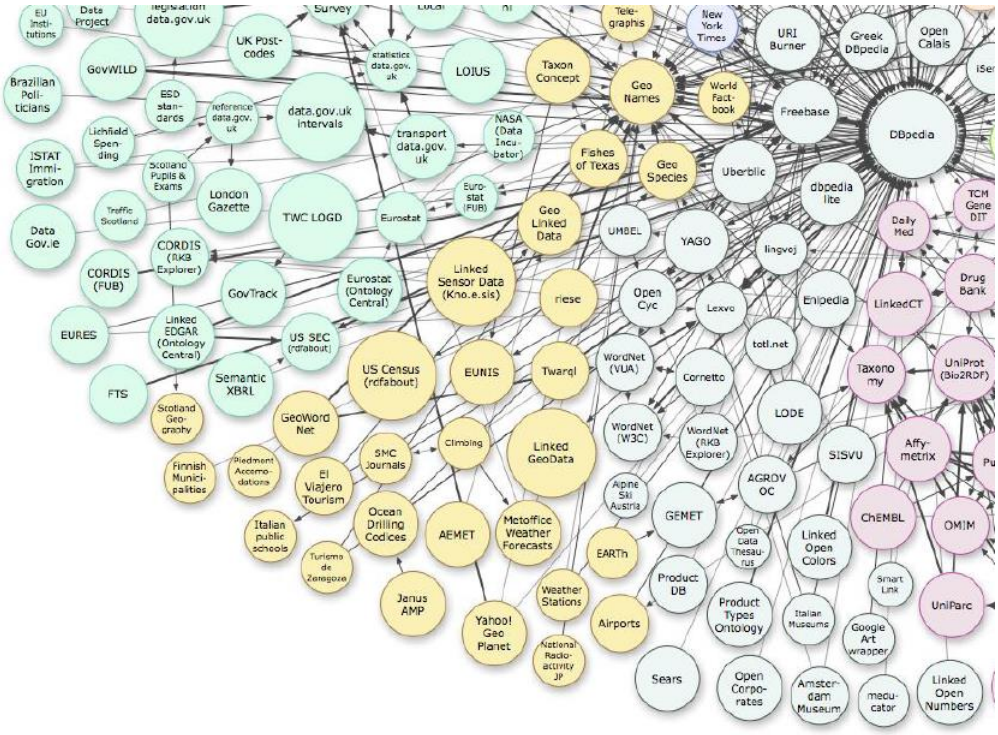


Figura 2. Parte geográfica del esquema *Linked Data*

3.4. *Open data* y *Open linked data*

Open data es una filosofía y práctica que persigue que determinada información esté disponible de forma libre para todo el mundo, sin restricciones de derechos de autor, de patentes u otros mecanismos de control. Está ética es aplicada también en otros ámbitos como el *software* libre, el código abierto o los estándares abiertos.

Se consideran abiertos aquellos datos que puedan ser utilizados, reutilizados y redistribuidos libremente por cualquier persona, y que se encuentra sujetos, cuando más, al requerimiento de atribución y de compartirse de la misma manera (CC-by-sa).

De la fusión de esta filosofía/iniciativa [28] y de la de datos enlazados, surge el Open Linked Data, que son datos abiertos en formato RDF (enlazados). Esto significa que el usuario puede enlazar datos provenientes de diversas fuentes, instituciones u organizaciones, explorar y combinar estos datos de manera libre y sin restricciones de *copyright*.

3.5. Principales iniciativas *Open Linked Data* de ámbito geográfico

En este apartado se van a citar las principales características de algunos proyectos de *Open Linked Data* que están operativos en la actualidad.

Algunos de ellos son trabajos de investigación, como el proyecto europeo SmartOpenData (apartado 3.5.1), otros son iniciativas libres como DBpedia (apartado 3.5.2) o Geonames Ontology (apartado 3.5.3) y las restantes se tratan de agencias comprometidas con la filosofía Open Linked data. Las agencias que se mencionarán serán Ordnance Survey (apartado 3.5.4), GeoLinkedData que toma datos del IGN (apartado 3.5.5), AEMETLinkedData, que toma datos de AEMET (apartado 3.5.6), y el Ayuntamiento de Gijón (apartado 3.5.7)

3.5.1. SmartOpenData

SmartOpenData [5], [6] es un proyecto europeo que tiene como objetivo la creación de una Infraestructura de datos enlazados y abiertos (incluyendo las herramientas de *software* e información) alimentados por fuentes de datos públicas y totalmente disponibles relacionadas con la protección e investigación ambiental en zonas rurales y en áreas protegidas europeas con sus parques nacionales. Este proyecto proporciona oportunidades para la pequeña y mediana empresa de generar nuevos e innovadores productos y servicios que pueden generar más negocios a la hora de la toma de decisiones para actuaciones políticas relacionadas con el medio ambiente. De esta forma pretende aumentar el valor de los datos a través de un lenguaje común de consulta que dará acceso a conjuntos de datos enlazados y disponibles en la nube del Open Linked Data.

Algunas de las características de los componentes del proyecto tienen como objetivo hacer posible:

- Un amplio acceso a información científica que permita realizar investigaciones en diferentes dominios de forma que permita la colaboración entre varios conjuntos de datos.
- El compromiso con la totalidad de nuevas formas de investigación científica y explorar la correlación entre los resultados de investigación.
- El uso de modelos, novedosas herramientas medioambientales e información de productos, basándose en estándares ampliamente aceptados.
- Proporcionar beneficios a investigadores, a agencias europeas, al sector industrial, a los políticos y a los ciudadanos que estén dentro del dominio medioambiental.
- La interoperabilidad entre diferentes repositorios de datos.

El proyecto tiene planeado contar con las siguientes interfaces:

- Un *endpoint* SPARQL que sea capaz de recibir consultas SPARQL de aplicaciones clientes y permita introducir parámetros reactividad, de recibir resultados de consultas SPARQL por parte de la envoltura de *endpoints* federados que forman el proyecto, enviar los resultados de las consultas SPARQL a los clientes, enviar las consultas SPARQL descompuestas y enviar los parámetros de reactividad a la envoltura de *endpoints* federados y al sistema de descomposición de consultas.

- Un sistema de descomposición de consultas que sea capaz de recibir consultas SPARQL de *endpoints* SPARQL y parámetros de reactividad, que tenga un sistema capaz de descubrir las fuentes de las consultas y que muestre consultas relevantes, con estadísticas de las instancias y cargado de información. Además debe poder enviar a todo a la envoltura de *endpoints* federados una lista de consultas SPARQL enlazadas con el *endpoint* correspondiente y recibir de ella una señal de control relativa a fuentes de información no disponibles u otras razones a tener en cuenta para una descomposición diferente.
- Un sistema para descubrir fuentes de información que sea capaz de recibir el tiempo de ejecución en el cargado de las información por parte de la envoltura de *endpoints* federados, de dar un sumario de consultas de información de acuerdo a patrones de consulta recibidos del sistema de descomposición de consultas, de consultar diferentes repositorios con alineaciones entre elementos y de ser examinado por el sistema de descomposición de consultas acerca de que fuentes de datos están disponibles.
- Una envoltura de *endpoints* federados que sea capaz de recibir del sistema de descomposición una lista de consultas SPARQL enlazadas con el *endpoint* correspondiente, recibir parámetros de reactividad SPARQL del *endpoint* SPARQL, enviar al sistema de descomposición una señal de control relativa a fuentes no accesibles, consultar repositorios con entidades alineadas conocidas, consultar los *endpoint* de la federación y enviar respuestas SPARQL a los *endpoints.*
- Un *endpoint* que haga sumario de la información y que esté formado por una herramienta que proporcione metadatos de cada fuente y que pueda ser consultada con fuentes RDF y pueda responder con *endpoints* relevantes.
- Un repositorio esquemático formado por una herramienta de alineación de ontologías y que pueda ser consultado sobre los alineamientos entre entidades.

Además de esto el proyecto pretende colaborar con el problema que supone el multilingüismo en la información geográfica, tratando, con la implementación del RDF, lograr una traducción más fácil de conceptos geográficos

También trabajará en la utilización de grandes cantidades de información en tiempo real.

La principal meta del proyecto es por tanto hacer que las infraestructuras INSPIRE, GMES y GEOSS más accesible a los ciudadanos, pero también a organizaciones públicas y privadas y a la pequeña y mediana empresa. Además de eso también pretende hacer que la información espacial Europea sea fácilmente reusable no sólo por expertos GIS sino también por el resto de ciudadanos. Pretende así identificar las posibilidades de establecimiento de conexiones semánticas entre las citadas infraestructuras y el contenido espacial enlazado del Linked Open Data, con el objetivo de generar un valor añadido, en particular dentro del dominio de la investigación ambiental.

3.5.2. DBpedia

El proyecto de DBpedia [7] es un proyecto colaborativo que pretende la extracción estructurada de la información de Wikipedia y hacerla accesible en la Web. Es un proyecto realizado por la Universidad de Leipzig, la Universidad Libre de Berlín y la compañía OpenLink Software.

DBpedia permite realizar consultas complejas contra Wikipedia, y enlazar diferentes conjuntos de datos de la Web con los datos de Wikipedia.

En la base de datos inglesa de DBpedia se describen 3,77 millones de entidades, entre ellas al menos 764 mil personas y 563 mil lugares. El contenido de la base de datos está disponible bajo licencia CC-BY-SA 3.0 y GFDL.

La información se almacena en formato RDF y se puede consultar la base de datos con el mencionado lenguaje SPARQL.

3.5.3. Geonames ontology

Geonames [8] es una base de datos geográfica gratuita y accesible a través de internet bajo licencia *Creative Commons 3.0*. Contiene más de 8 millones de nombres geográficos que corresponden a más de 6,5 millones de lugares existentes.

Una de las iniciativas de Geonames es Geonames Ontology, la cual hace posible añadir información semántica geoespacial a la web. En este proyecto se representa a cada característica a través de una URI estable. Esta URI proporciona acceso, mediante transferencia de información, a un Wiki en página HTML o a una descripción de recursos RDF.

Algunas de las características que implementa la ontología de Geonames es que puede proporcionar las divisiones administrativas de un determinado lugar (*contains*), los países vecinos a otro (*neighbours*) o los lugares cercanos a un punto (*nearby*). Por ejemple la URL para obtener en un documento RDF los países vecinos de Francia es:

http://sws.geonames.org/3017382/neighbours.rdf

Las formas de acceder a la ontología de Geonames son las siguientes:

- A través del visualizador "*mother earth*".
- El servicio "*search*" de Geonames introduciendo como "type=rdf"
- Descargando el un acceso a la base de datos y accediendo a través del patrón http://sws.geonames.org/geonameId/
- Descargando directamente toda la base de datos en RDF que ocupa aproximadamente 2Gb.

3.5.4. Ordnance Survey

Ordnance Survey [9] es la agencia nacional cartográfica de Gran Bretaña y uno de los mayores editores de mapas del mundo. Entre otras iniciativas, también se ha interesado recientemente en formar parte de la web semántica.

Como primer paso decidieron producir un *"gazzetter"* de unidades administrativas del Reino Unido. A cada región se le determinó un identificador único en forma de URI y se le asignaron ciertos datos como su nombre o relación espacial que tiene con otras regiones. Un ejemplo sería el siguiente URI: http://data.ordnancesurvey.co.uk/id/7000000000037256

La Wiki que proporciona cada URI es más estética que las que normalmente se pueden encontrar, pero no cuenta explícitamente con los códigos a utilizar en las consultas SPARQL.

Además de eso también ha publicado URI's para cada código postal en Reino Unido y enlazado estas URI's con las regiones administrativas.

El siguiente paso fue crear un conjunto mayor de información espacial. Dicha información incluye asentamientos, accidentes geográficos, hidrografía y divisiones administrativas. En la actualidad se está creando *linked data* a partir de divisiones históricas y fuentes de información de los distritos para proveer de contexto temporal a nombres de lugar actuales.

3.5.5. GeoLinkedData

GeoLinked Data (.es) [10] es una iniciativa abierta del Ontology Engineering Group (OEG) [22] de la Universidad Politécnica de Madrid (UPM) destinada al enriquecimiento de la Web Semántica con datos geoespaciales del territorio nacional español. Esta iniciativa se ha puesto en marcha con la publicación de diversas fuentes de información procedentes del Instituto Geográfico Nacional, haciéndolas disponibles como bases de conocimiento conforme a los principios de Linked Data. Además, estos datos se interrelacionan con otras bases de conocimiento existentes en la iniciativa Linking Open Data. De esta manera, España se suma a la iniciativa que otros países como Reino Unido y Alemania han comenzado recientemente.

3.5.6. AemetLinkedData[1]

AEMET es la Agencia Estatal de Meteorología. Su objetivo básico es la prestación de servicios meteorológicos, que sean competencia del Estado.

AemetLinkedData [11] es una iniciativa del citado Ontology Engineering Group (OEG) [22] destinado a la publicación de datos meteorológicos del territorio español en formato RDF y conforme a los principios de Linked Data.

3.5.7. Ayuntamiento de Gijón

[1] Los últimos datos disponibles son del 19 de septiembre de 2011, cuando el FTP de AEMET todavía era público.

El Ayuntamiento de Gijón [12] se ha unido a la iniciativa de facilitar públicamente los datos y de forma enlazada y expone muchos de sus conjuntos de datos en formato RDF. Cuenta además con un SPARQL *endpoint*[2].

De esta forma, además de publicar los datos con semántica en formato "Linked Data" (Tecnología Web 3.0 o Web Semántica), se facilita una herramienta para la consulta.

3.6. *Frameworks* para manejo de información enlazada

En este apartado se van a analizar algunos de los motores de representación de datos RDF y de visualización o consumo de este tipo de datos. Los que se han podido documentar son 4Store, OpenLinkVirtuoso, Sesame, Oracle Spatial 11g, AllegroGraph y Jena2 [29], [32], [33].

3.6.1. 4Store

4Store [13] es un almacén de datos RDF/SPARQL en lenguaje C y que tiene como objetivo principal el desarrollo de aplicaciones web semánticas, permitiendo consultar los datos enlazados almacenados por los usuarios. Sus principales características son:

- Soporta RDF
- Tiene licencia GPL.
- Trabaja bajo sistema operativo Linux, Unix, Mac , y Windows
- Soporta SPARQL

3.6.2. Virtuoso

Virtuoso [14] es un Data Store híbrido que combina las funcionalidades de los gestores de bases de datos enlazados (RDBMS), de los sistemas gestores de bases datos objeto-relacional (ORDBMS), bases de datos virtuales, RDF, XML y aplicaciones web. Entre las características más relevantes se pueden citar las siguientes:

- Posee licencia pública general (GPL) para el producto OpenLinkVirtuoso y también licencia pagada para otras versiones del producto.
- Posee un diccionario de datos en donde se almacena toda la información de los objetos de los usuarios.
- Trabaja en los sistemas operativos Windows, Mac, Linux y Unix.
- Soporta los formatos RDF y XML.
- Soporta SPARQL
- Soporta ISQL, lenguaje que permite una fácil administración y mantenimiento de la base de datos.
- Mecanismos de seguridad basados en privilegios y roles.
- Provee conexiones a fuentes de datos en formatos, XAML, ODBC, JDBC, ADO.NET y OLE.DB.

[2] No está operativo.

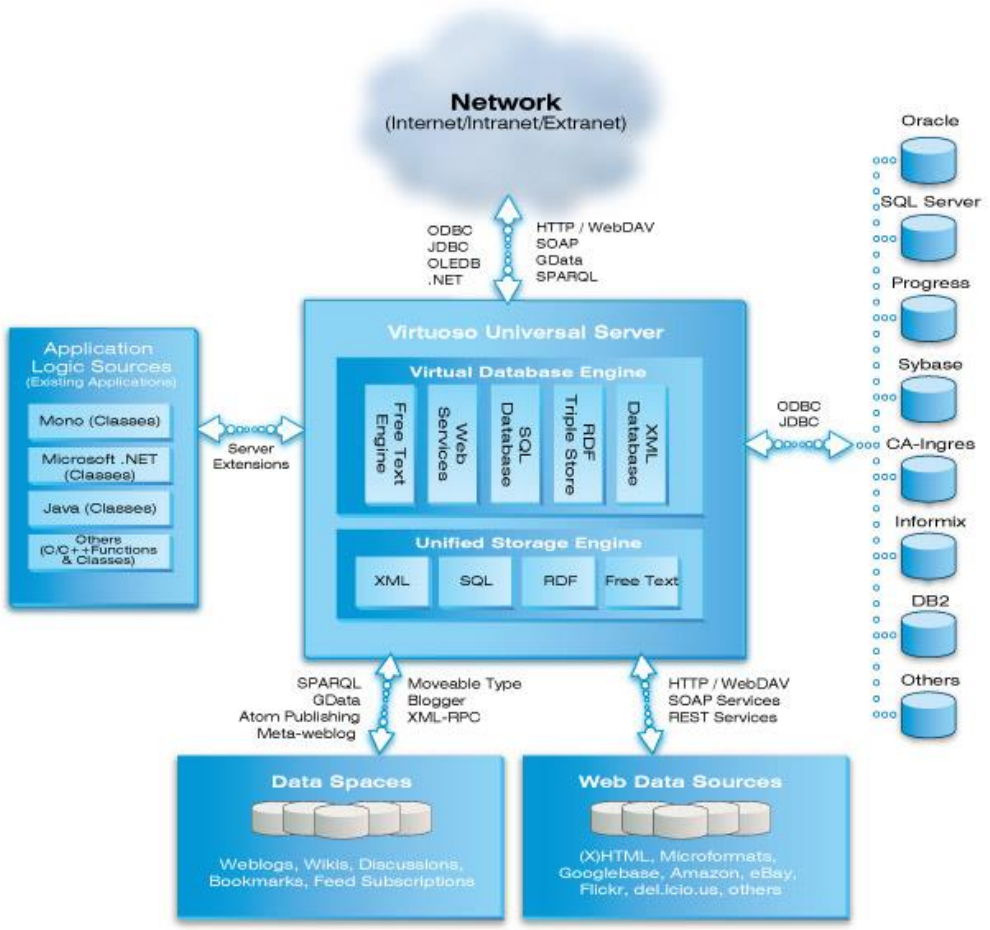


Figura 3. Esquema de funcionamiento de Virtuoso

3.6.3. Sesame

Sesame [15] es una plataforma diseñada en java, que tiene como objetivo el almacenamiento y consulta de datos enlazados en RDF y RDF Schema. Se puede utilizar como base de datos RDF y RDFs o como una librería para aplicaciones que necesiten utilizar datos RDF internamente. Sus características son:

- Puede ser implementado en sistemas de almacenamiento como base de datos relacionales, sistemas de ficheros e indexadores de palabras claves.
- Soporta los formatos RDF y RDFS.
- Soporta los lenguajes de consulta SeRQL y SPARQL.
- Ofrece varias herramientas de análisis, interpretación, consulta y almacenamiento de información enlazada.
- Trabaja en los sistemas operativos Windows, Unix, Solaris, Mac e Irix.
- Es de código abierto.
- Las tripletas en Sesame se les asocia un contexto, es decir, en vez tripletas son cuádruplas.

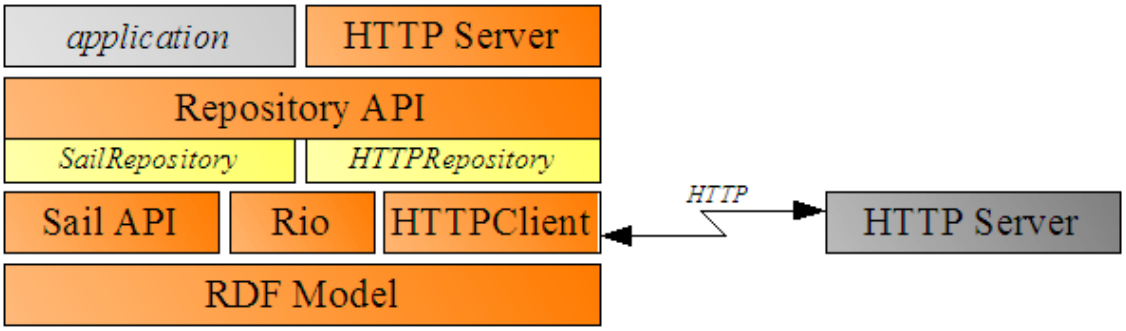


Figura 4. Niveles del esquema de funcionamiento de Sesame

3.6.4. Oracle Spatial 11g

Oracle Spatial 11g [16] contiene un *framework* abierto de gestión de RDF escalable, seguro y confiable, basado en un modelo de datos de grafos. Las tripletas RDF están indexadas y la consulta es similar a otros tipos de datos de bases relacionales. Sus características principales son:

- Toda la información puede ser consultada con SQL, al que se añade la función embebida SEM_MATCH, que cubre la mayoría de funcionalidades de SPARQL.
- Introduce un motor de inferencia nativa para razonamientos sobre subconjuntos de OWL.
- Soporta SQL
- Soporta los formatos RDF y RDFS
- Trabaja sobre las plataformas Windows, Linux, Mac y Solaris
- Las tripletas tienen asociado un identificador único.

3.6.5. AllegroGraph

AllegroGraph [17] es un sistema de carga y consulta de datos RDF, desarrollada para sistemas de 64 bits, persistente y de alto rendimiento, que contiene una interfaz SPARQL y un servidor RDF. Algunas de sus características son:

- Soporta SPARQL
- Se pueden manipular los datos en tripletas en diferentes interfaces y lenguajes como JAVA, HTTP y LISP.
- Soporta razonamiento RDFS++ y Prolog.
- Incorpora un cliente java mejorado.
- Carga de archivos en formato Turtle.

3.6.6. Jena2

Jena2 [18] es un *framework* desarrollado en java que tiene como objetivos la construcción de aplicaciones para la web semántica y proveer un ambiente de programación para RDF, RDFS y OWL. Además de esto, dispone de un motor de inferencias basado en reglas, lo que lo convierte en un modelo ideal para cualquier proceso automatizado de creación de

contenidos destinados a ser usados en canales de información. Algunas de sus características son:

- Posee un API para RDF que soporta creación, manipulación y consulta de RDF.
- Permite realizar lectura y escritura de documentos en formato RDF/XML, N3 y N-Triples.
- Posee un API para OWL.
- Almacenamiento persistente y en memoria.

3.7. Librerias de *JavaScript*

En este apartado se van a describir las librerías de JavaScript que se han empleado en el desarrollo de la aplicación del presente estudio.

3.7.1. Openlayers

OpenLayers [19] es una librería de Javascript de código abierto para mostrar mapas interactivos en navegadores web. Ofrece un API para acceder a diferentes fuentes de información cartográfica en la red: Web Map Services, Mapas comerciales (Google Maps, Bing, Yahoo), Web Features Services, distintos formatos vectoriales y mapas de OpenStreetMap, entre otras posibilidades.

Esta plataforma permite colocar un mapa dinámico en cualquier página web de forma fácil, puede mostrar teselas y marcadores cargados desde cualquier fuente.

En su página web cuenta con un gran número de ejemplos de explotación de su servicio que facilita el trabajo al desarrollador, además de una documentación muy completa tanto de su API como orientada al usuario.

Han sido muchas las versiones de OpenLayers que se han ido publicando y que han ido haciendo a este software cada vez más completo y flexible, siendo la más reciente la versión 2.13.1. En la actualidad está a punto de ser lanzada la versión 3, que incluye ciertos avances como la mejora del tratamiento de datos vectoriales o la mayor sencillez a la hora de personalizar los mapas con hojas de estilos (CSS), entre otras.

Otro punto positivo de OpenLayers es la gran cantidad de información relacionada con la resolución de problemas de este *software* en foros y páginas especializadas.

3.7.2. jQuery

jQuery [20] es una librería ligera de JavaScript creada por John Resig que sigue la filosofía de “escribe menos, haz más”. Su propósito es hacer más fácil usar JavaScript en las páginas web.

Esta librería transforma un gran número de tareas que requieren muchas líneas de código en JavaScript, en métodos que se pueden llamar con una simple línea de código. También simplifica otras partes más complicadas de JavaScript como son las llamadas con Asynchronous JavaScript And XML (AJAX) o la manipulación con Document Object Model (DOM).

Existen muchas otras librerías de JavaScript semejantes a JQuery, pero esta es probablemente la más popular y extensible de todas.

jQuery es *software* libre y de código abierto y permite su uso en proyectos libres y privados. Consiste en un único fichero JavaScript que contiene las funcionalidades comunes de DOM, eventos, efectos y AJAX. La característica principal de la biblioteca es que permite cambiar el contenido de una página web sin necesidad de recargarla, mediante la manipulación del árbol DOM y peticiones AJAX. Para ello utiliza las funciones $() o jQuery().

3.8. AJAX

AJAX [21] es una técnica de desarrollo web para la creación de páginas web rápidas y dinámicas. Permite a las páginas web ser actualizadas de forma asíncrona para intercambiar pequeños volúmenes de datos con otro servidor. Esto hace posible actualizar partes de una página web, sin recargar toda la página.

Las aplicaciones que cuentan con AJAX en su código se ejecutan en el cliente, es decir, en el navegador, mientras se mantiene la comunicación asíncrona con el servidor en segundo plano.

Es por tanto una combinación de 4 tecnologías existentes:

- HTML y hojas de estilos en cascada (CSS)
- Document Object Model (DOM).
- El objeto XMLHttpRequest para intercambiar datos con el servidor de forma asíncrona.
- El formato de intercambio de datos es generalmente XML, pero también acepta texto plano y JSON entre otros.

4. Metodología

Existen tres partes claramente diferenciadas en el desarrollo del presente trabajo. Una primera parte en la que se explotan datos enlazados a partir de fuentes y servidores externos, otra segunda parte donde se trabaja en un servidor propio y con datos enlazados personales y otra en la que se integran en una misma aplicación datos provenientes de los tres modos de trabajo (Figura 5) [31].

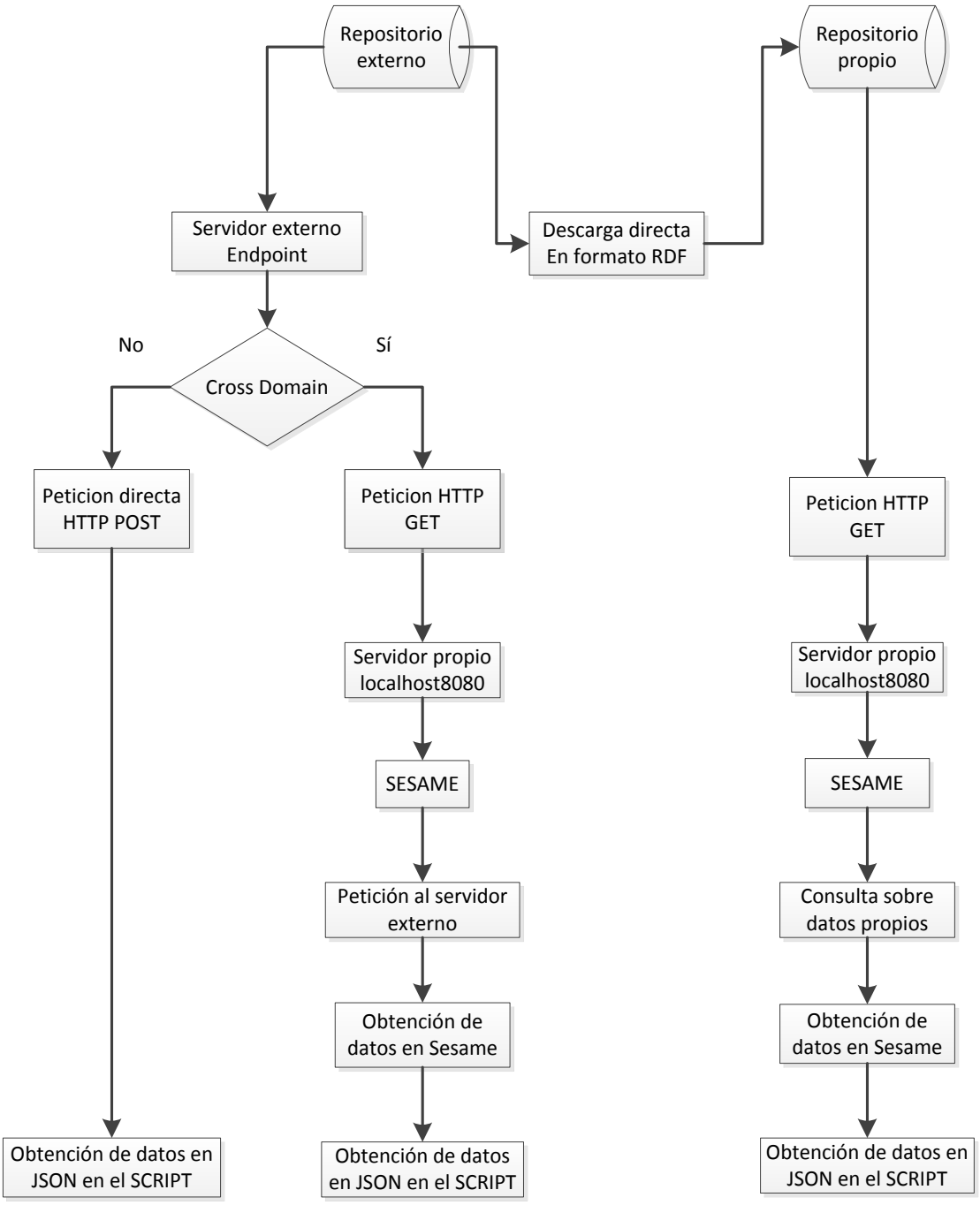


Figura 5. Esquema de trabajo

Se utilizará el servidor SPARQL de DBpedia [3](*endpoint*) para realizar la explotación de datos enlazados de una fuente externa con la ayuda de peticiones HTTP.

Así mismo, para comprobar la capacidad de la base de datos de DBpedia, se realizarán una serie de consultas que permitan comprobar la utilidad de los datos enlazados, ya que la cantidad de información que esta contiene lo permite.

Para realizar la explotación de datos enlazados de una fuente externa a través de un *software* especializado en tratamiento de datos enlazados (Sesame), se utilizarán los *endpoints* SPARQL de las iniciativas AEMETLinkedData y GeoLinkedData del Ontology Engineering Group.

El *software* Sesame tiene la posibilidad de registrar como repositorio cualquier *endpoint* SPARQL y explotarlo con consultas al servidor propio sobre el que esté instalado. Es decir, Sesame, en este caso, trabaja como un nexo de unión entre el cliente y el servidor de datos enlazados.

En cuanto a la explotación de datos geográficos enlazados propios, se utilizaran los datos enlazados disponibles en la página web del Ayuntamiento de Gijón. Dentro de Sesame se creará un repositorio a partir de varios documentos RDF que estén almacenados en el equipo y se podrá consultar con la ayuda del lenguaje SPARQL y a través de peticiones HTTP.

Todos estos procesos de obtención de datos se integrarán en una sola aplicación programada en JavaScript, que tendrá las siguientes funcionalidades:

- Un menú de consulta directa SPARQL, capaz de enviar peticiones HTTP POST al *endpoint* de DBpedia, con la consulta introducida por el usuario. Tras enviar la consulta, la aplicación recibirá el resultado de la misma en formato JavaScript Object Notation(JSON), será capaz de procesarlo y de ilustrarlo en un mapa interactivo con la ayuda de marcadores. (Apartado 6.2.2).
- Un menú interactivo en el que se permita consulta determinados fenómenos geográficos como ciudades, montañas o volcanes... y filtrarlos en función de su altura. Será capaz de crear la consulta SPARQL en función de la elección del usuario en la interfaz, enviarla a través de petición POST al *endpoint* DBpedia, de recibir el resultado en JSON, procesarlo y representar los datos en el visualizador geográfico. (Apartado 6.2.3)
- Un menú interactivo en el que se permita consultar asentamientos de población y filtrarlos en función de la población. Al igual que en el caso anterior, el programa creará la consulta en función de las elecciones del usuario, la enviará al *endpoint* de DBpedia con una petición HTTP POST, recibirá el JSON de resultado y lo procesará y finalmente representará los datos en el visualizador.(Apartado 6.2.4)
- Un menú interactivo en el que se realice la consulta de todas las estaciones disponibles en el proyecto AEMET LinkedData, y que además de eso permita consultar la información de alguna variable climatológica de alguna estación. Para lograr esto, enviará las consultas creadas interactivamente por el usuario a través de petición HTTP GET al repositorio creado en SESAME, este a su vez enviará la petición al *endpoint* y

[3] http://dbpedia.org/sparql

traerá de vuelta el JSON con el resultado de la consulta. La aplicación procesará este JSON y lo representará en el visualizador, mostrando en el *PopUp* correspondiente la información de las variables que se pidan. (Apartado 6.2.5)

- Un menú interactivo que explotará los datos geográficos enlazados del Ayuntamiento de Gijón almacenados en un repositorio personal. Será capaz de crear la consulta SPARQL según la elección del usuario y enviarla a través de una petición HTTP GET al servidor propio. Debe recibir como respuesta un JSON con los datos que se pedían en la consulta, ser capaz de procesar el JSON y de representarlo en el visualizador de mapas. (Apartado 6.2.6)
- Un menú interactivo capaz de explotar algunos de los datos enlazados del proyecto GeoLinkedData. Debe crear, al igual que los anteriores menús, la consulta en función de la elección del usuario y enviarla por petición HTTP GET a Sesame, donde estará registrado el repositorio de GeoLinkedData. Sesame conseguirá los datos tras consultar el *endpoint* y los devolverá en forma de JSON a la aplicación. Una vez obtenidos los datos se deberán procesar y representar en el mapa. (Apartado)
- Un módulo que fije los marcos de trabajo de todos los menús. Este módulo permitirá la captura interactiva mediante clic de coordenadas, las cuales, las almacenará de forma que puedan ser utilizadas para filtrar las consultas SPARQL y así restringir el ámbito de trabajo. Será capaz de trabajar con tres marcos de trabajo independientemente. (Apartado 6.2.7)

5. Desarrollo interno de la aplicación

En este apartado se va a mostrar cómo se ha logrado la obtención de información espacial de diferentes repositorios de datos enlazados a través de diferentes metodologías. La primera de ellas se realizará con petición directa HTTP a un servidor externo, la segunda de ellas se hará del mismo modo una petición HTTP, pero utilizando el software Sesame mencionado en el apartado 3.6.3 como nexo de unión y la tercera de ellas se hará también por petición HTTP pero a un servidor privado donde estén almacenados datos propios.

En la Figura 6 aparece un esquema que ilustra lo anterior.

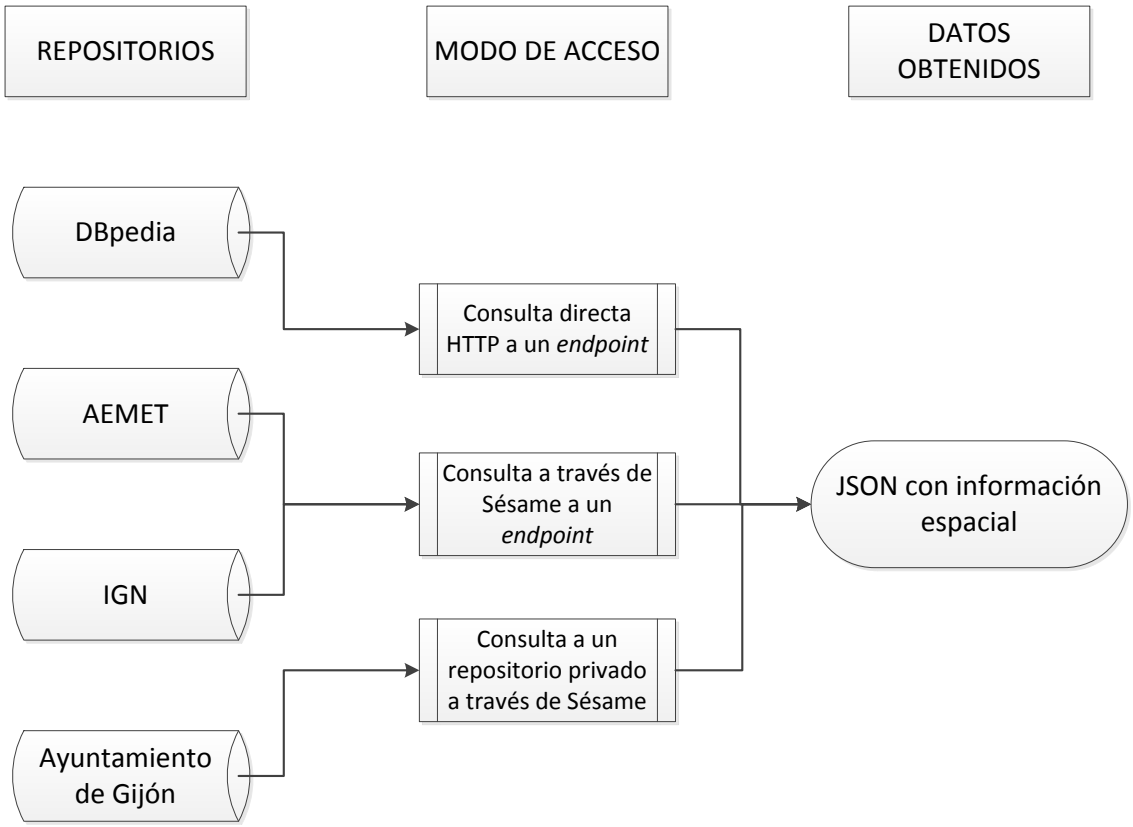


Figura 6. Esquema de trabajo interno de la aplicación

5.1. Extracción de información de un servidor externo de modo directo

Para explotar datos RDF a través de internet se requiere primeramente identificar si un determinado organismo facilita este tipo de datos y seguidamente, el modo en que lo hace. La forma de proporcionarlos puede ser mediante descarga o mediante la utilización de un servidor propio de la organización (*endpoint*) que los suministre.

Algunos de los organismos a nivel nacional que proveen información geográfica enlazada a través de un *endpoint* son la Agencia Estatal de Meteorología (AEMET[4]) o el Institutito Geográfico Nacional (Geo-Linked Data[5]) gracias a la labor del OEG. A nivel

[4] http://aemet.linkeddata.es/sparql.html
[5] http://linkedgeodata.org/OnlineAccess/SparqlEndpoints

internacional DbPedia[6] o GeoNames[7] son las más destacables y ampliamente utilizadas. Así mismo, se han encontrado diversos organismos a nivel local que cuentan con un *endpoint*. Es el caso del Ayuntamiento de Gijón[8], el de Zaragoza o el de Barcelona. De estos últimos el único que proporciona datos geográficos es el ayuntamiento de Gijón.

Para contactar con el servidor se debe recurrir al protocolo HTTP. Este protocolo basado en peticiones y respuestas permite la comunicación entre un cliente y un servidor a través de sus métodos GET o POST. El primero de ellos envía la petición a una fuente determinada con toda la información visible al usuario en la URL y el segundo envía la petición a la fuente con toda la información oculta[9] al usuario y con formularios.

Para contactar con el servidor de las citadas organizaciones se envía en forma de URL con una raíz que identifica al servicio. Las raíces de los servicios utilizadas en este estudio [34] son las que aparecen reflejadas en la **¡Error! No se encuentra el origen de la referencia.**.

Tabla 1 Raíces del endpoint de consultas a repositorios de datos enlazados

Organismo	Raíz del Endpoint
Dbpedia	http://dbpedia.org/sparql
AEMET	http://aemet.linkeddata.es/sparql
Geo-Linked data	http://geo.linkeddata.es/sparql
Ayuntamiento de Gijón	http://datos.gijon.es/sparql (No operativo)

Para completar la consulta, en la misma URL se debe proporcionar justo después de la raíz, la consulta SPARQL que se va realizar a los datos del organismo correspondiente. Finalmente también se proporciona el formato de recepción de los datos que se piden, y se especifica al final del enlace URL. Un ejemplo de consulta a un *endpoint* sería el siguiente:

URL que devuelve un JSON:

http://dbpedia.org/sparql?query=SELECT+?name+?lat+?lon++WHERE+{?c+a++yago:ProvincesOfSpain+.++rdfs:label+?name.+geo:lat+?lat+.+geo:long+?lon+.+FILTER+(+LANG(?name)+=+”es”+&&+(?lat%3E+39.29508523669101+&&+?lat+=+43.12814412610149%29+&&(?lon=+-8.57462158203123++&&+?lon+=+-0.13712158203126765))}&format=json

De esta forma se pueden explotar datos enlazados disponibles en un servidor externo.

Al estar desarrollando una aplicación para la Web, cabe mencionar la existencia de la protección de dominio cruzado (*Cross Domain*). El *Cross Domain* es un mecanismo de seguridad de las comunicaciones en navegadores actuales. Evitan que un script o una aplicación de una página web puedan acceder a un servidor web diferente del que residen. Esto se hace para evitar el sabotaje y la suplantación de identidad en internet.

[6] http://dbpedia.org/About
[7] http://www.geonames.org
[8] https://datos.gijon.es/page/12217-servicio-sparql
[9] Existen formas de mostrar esta información al usuario.

El único organismo que no tiene protección *Cross Domain* de los consultados en este estudio es DBpedia[10], todos los demás tienen este tipo de protección.

5.2. DBpedia

Los datos enlazados de DBpedia se pueden consultar de dos formas, una de ellas es a través de *Wikis*, en las que se muestran todos los datos relacionados con un concepto, y que están asociados a una URI, por ejemplo, la URI que identifica el concepto Madrid es: http://dbpedia.org/page/Madrid y la Wiki asociada aparece en la Figura 77.

Figura 7. Wiki de DBpedia referente a Madrid

La otra forma es a través de su servidor de datos enlazados o *endpoint* [24], el cual permite obtener datos a través de una consulta SPARQL y con un formato concreto. Los formatos que contempla dicho servidor son HTML, *SpreadSheet*, XML, JSON, JavaScript, Turtle, RDF/XML, N-Triples, CSV y TSV.

En el caso de DBpedia, el *endpoint* no sufre una protección estricta contra el *Cross Domain*, con lo cual se pudo realizar la petición directamente, como se mostrará en los apartados 5.2.1 y 5.2.2.

5.2.1. Consultas realizadas a DBpedia y resultados

Para mostrar las posibilidades de DBpedia se llevaron a cabo las siguientes consultas:

- Nombre, latitud y longitud de las provincias de España, que estén registradas como tal en DBpedia con la propiedad yago:ProvincesOfSpain . Además de eso, filtra los resultados para obtener sólo los nombres de las provincias en español.

```
SELECT ?name ?lat ?lon
WHERE {?c a       yago:ProvincesOfSpain   ;
```

[10] Se ha experimentado cierto comportamiento errático en la disponibilidad del Endpoint, mostrando durante algunos períodos de tiempo respuestas a las peticiones que adolecían de restricciones de Cross Domain.

```
        rdfs:label ?name;
        geo:lat ?lat ;
        geo:long ?lon . FILTER ( LANG(?name) = 'es')}
```

Como se trabajó con formato JSON, la devolución de esta consulta tendrá la siguiente forma:


```
"head": { "link": [], "vars": ["name", "lat", "lon"] },
  "results": { "distinct": false, "ordered": true, "bindings": [
    { "name": { "type": "literal", "xml:lang": "es", "value":
"\u00C1lava" } , "lat": { "type": "typed-literal", "datatype":
"http://www.w3.org/2001/XMLSchema#float", "value": "42.8445" },
"lon": { "type": "typed-literal", "datatype":
"http://www.w3.org/2001/XMLSchema#float", "value": "-2.76033" }},
    { "name": { "type": "literal", "xml:lang": "es", "value":
"Cantabria" }  , "lat": { "type": "typed-literal", "datatype":
"http://www.w3.org/2001/XMLSchema#float", "value": "43.3333" },
"lon": { "type": "typed-literal", "datatype":
"http://www.w3.org/2001/XMLSchema#float", "value": "-4.0" }}…
```


- Nombre, latitud, longitud de las capitales de provincias de Ecuador, las ciudades de Illinois o las ciudades más importantes de Francia:

Se sustituye en la consulta anterior la etiqueta yago:ProvincesOfSpain por las etiquetas yago:ProvincialCapitalsInEcuador, yago:CitiesInIllinois y yago:CitiesInFrance respectivamente.

- Nombre, latitud, longitud y población de las capitales de Europa, filtrando aquellas que tengan una población inferior a 3000000:

```
SELECT ?name ?long ?lat ?pob
WHERE {

        ?c a yago:CapitalsInEurope ;
         rdfs:label ?name ;
         geo:long   ?long ;
         geo:lat    ?lat ;
         dbpedia-owl:populationTotal    ?pob.
         FILTER( lang(?name)='es' && (?pob< 3000000) )
       }
```

- Nombre, latitud, longitud y población de los municipios de la provincia de Lugo, de las provincias de España, de las capitales de Sudamérica, de las capitales de Oceanía, de los elementos de DBpedia que estén etiquetados como ciudad y las capitales de Asia, filtrando aquellas que tengan una población inferior a 3000000:

Se sustituye en la consulta anterior la etiqueta yago:CapitalsInEurope por las etiquetas yago:MunicipalitiesInTheProvinceOfLugo, yago:ProvincesOfSpain,

yago:CapitalsInSouthAmerica, yago:CapitalsInOceania, dbpedia-owl:City, y yago:CapitalsInAsia, respectivamente.

- Nombre, latitud, longitud y altura de las capitales de los estados de Estados Unidos, filtrando aquellas que tengan una altura superior a 1000 m.:

```
SELECT ?name ?lon ?lat ?alt
WHERE { ?c rdf:type yago:StatesOfTheUnitedStates .
?c dbpedia-owl:capital ?objeto.
?objeto rdfs:label ?name.
?objeto  geo:long   ?lon .
?objeto geo:lat    ?lat .
?objeto dbpedia-owl:elevation
?alt. FILTER(LANG(?name) = 'es' && (?alt > 1000))}
```

Es importante recalcar que en DBpedia cada estado de Estados Unidos no tiene almacenado la altura de la capital, con lo cual se debe primero obtener cual es la capital de cada estado con la etiqueta dbpedia-owl:capital y después obtener el valor de elevación que tiene asociado. **En esta consulta se puede observar el potencial del Linked Data**.

- Nombre, latitud, longitud y altura de las capitales de provincia de Ecuador, de las capitales de estado de Alemania, de las capitales de Europa, de los elementos etiquetados como ciudad, los volcanes activos y las montañas, filtrando aquellas que tengan una altura superior a 1000 m.:

```
SELECT ?name ?lon ?lat ?alt
WHERE { ?objeto rdf:type yago:ProvincialCapitalsInEcuador.
?objeto rdfs:label ?name.
?objeto  geo:long   ?lon .
?objeto geo:lat    ?lat .
?objeto dbpedia-owl:elevation
?alt. FILTER(LANG(?name) = 'es' && (?alt > 1000))}
```

Con esta consulta se obtienen las capitales de provincial de Ecuador que tengan una elevación superior a 1000 m. Para obtener los otros elementos citados, se debe substituir el valor de yago:ProvincialCapitalsInEcuador por los de yago:GermanStateCapitals para las capitales de estado alemanas, dbpedia-owl:City para los elementos etiquetados como ciudad, yago:CapitalsInEurope para las capitales de estado en Europa, y dbpedia-owl:Mountain para las montañas. En el caso de los volcanes se debe substituir rdf:type yago:ProvincialCapitalsInEcuador por dcterms:subject category:Active_volcanoes.

Para comprobar el funcionamiento de las anteriores consultas se utilizó tanto la interfaz que proporciona el *endpoint* de DBpedia (http://dbpedia.org/sparql), como por cuestiones de comodidad, la aplicación Sesame, que será citada y debidamente explicada en el apartado 4.2.2.

Cabe decir que toda **la información geográfica extraída de los repositorios tiene un carácter puntual**, ya que en ninguna de las fuentes de información se han encontrado datos geográficos con un tipo de entidad diferente al punto.

5.2.2. Explicación del proceso

Para conseguir obtener la información a través de la petición HTTP se creó una función en JavaScript capaz de hacerlo. A dicha función se le deben pasar cuatro variables, una que almacene un texto con la consulta, otra que almacene otro texto (*string*) con la raíz del *endpoint*, otra que llame a una función que procese los datos JSON que se obtengan y una última que indique si la consulta esta depurada o no.

La función crea primeramente una variable que almacena la consulta, fija el tipo de petición a realizar, el servidor que se va a consultar y el formato de salida de los datos con las siguientes sentencias en AJAX:

- xmlhttp.open('POST', *endpoint*, true): Envía el tipo de petición y a que servidor es realizada (la raíz del *endpoint*).
- xmlhttp.setRequestHeader("Accept", "application/sparql-results+json") : Establece el formato de salida de los datos.
- xmlhttp.send(querypart) : Envía la consulta a realizar.

5.3. Extracción de información de un servidor externo con la ayuda de Sesame

Tras repasar las características de todos los sistemas mencionados anteriormente y revisado algunos estudios [29], [32], [33], se ha llegado a la conclusión de que no es posible realizar una comparación exhaustiva de uno con otro, sino que se puede seleccionar alguno en función de las necesidades del usuario. De este modo, se observó, por ejemplo, que Sesame posee un tiempo de respuesta ante determinadas búsquedas más rápido que otros sistemas como Jena, pero este posee hasta el triple de capacidad de almacenamiento. En cuanto a los sistemas propietarios (AllegroGraph, Virtuoso y Oracle 11g), se observa una mayor potencia y escalabilidad, ya que superan los 1000 millones de sentencias, y los anteriores no pasan de los 200.

Sopesando las necesidades de este trabajo, se ha decidido trabajar con Sesame, ya que es uno de los más rápidos a la hora de las consultas, lo que permitirá practicar las consultas de una forma más ágil. Aunque no tenga la mayor de las capacidades entre sus competidores para almacenar sentencias, para este trabajo no es necesaria.

Además de esto, Sesame tiene todo su código abierto y disponible, lo que casa con la filosofía del Open Linked Data. Su interfaz es muy intuitiva con un modo consola, cuenta con aplicación *Web Client* y trabaja con ficheros almacenados en memoria y con archivos externos.

Como se ha citado en el apartado 3.6.3, Sesame es un *framework* estándar de facto para procesar datos en RDF, lo que incluye analizar, almacenar soluciones, razonamiento y consulta, utilizando lenguaje SPARQL. Fue desarrollado por la comunidad OpenRDF, que además del citado *framework*, cuenta con otro de nombre AliBaba. Esta plataforma es una librería para desarrollar aplicaciones más complejas de almacenamiento de datos en RDF. Consiste en una colección de módulos que proveen a Sesame de abstracciones simplificadas de almacenes RDF para acelerar el desarrollo y facilitar el mantenimiento de aplicaciones.

La forma de trabajar de Sesame [25], [26] aparece sintetizada en la Figura 8.

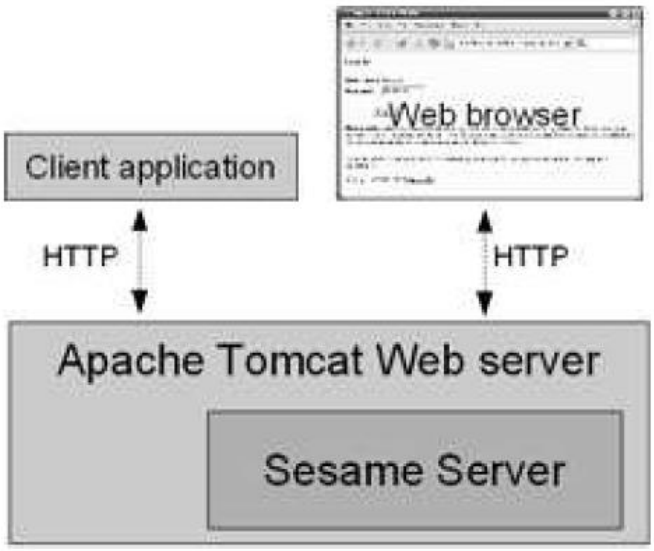


Figura 8. Esquema de funcionamiento de Sesame

Como se puede observar en la figura anterior, el *framework* Sesame aparece montado en el servidor de Apache Tomcat Web server. Una vez hecho esto, a través de peticiones HTTP se puede consultar al Sesame a partir de una aplicación cliente o de un navegador web. Este proceso se ilustrará más en profundidad en los siguientes apartados.

5.3.1. Instalación de Sesame

Para lograr la correcta instalación de Sesame [27], lo primero que se debe hacer es crear un servidor propio en el equipo. De la variedad de servidores disponibles que se podrían elegir (Jetty, Tomcat, IIS…), se decidió utilizar el *software* servidor Apache-Tomcat en su versión 6.0.41, por ser el que tiene mejor documentación disponible.

Una vez descargado de su página web el archivo en formato ZIP que contiene el *software* (http://tomcat.apache.org/download-60.cgi), se debe descomprimir y copiar todos los documentos en una carpeta situada en el directorio que se vaya a trabajar[11] (Figura 9).

[11] Se recomienda que sea en el directorio C:\ para facilitar su acceso.

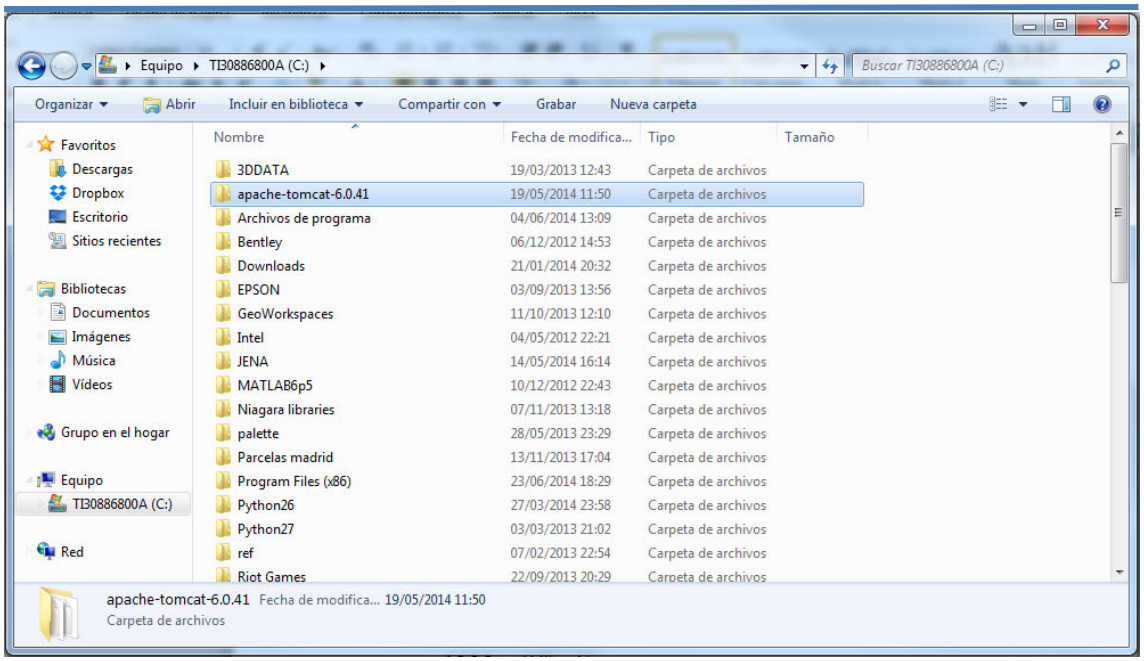


Figura 9. Directorio de instalación del servidor TomCat

De esta forma ya está instalado el servidor en el equipo.

El siguiente paso es descargar el *software* Sesame de la página web de OpenRDF (http://sourceforge.net/projects/sesame/files/Sesame%202/2.7.12/) en formato ZIP. Una vez descargado, se descomprime. Dentro de la carpeta se encuentran varios ficheros, de los cuales se deben tener en cuenta los siguientes:

- La API para peticiones HTTP: openrdf-sesame.war
- La interfaz de trabajo: openrdf-workbench

Estos ficheros se deben copiar y pegar en el directorio C:\apache-tomcat-6.0.41\webapps (Figura 10):

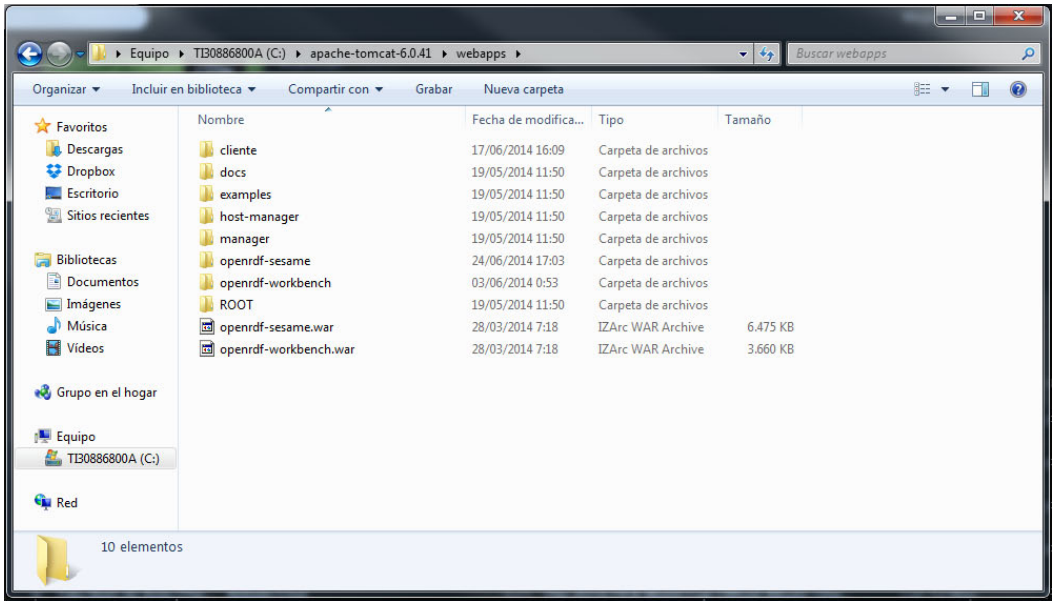


Figura 10. Directorio de instalación de Sesame sobre TomCat

En este punto del procedimiento, la instalación del *framework* Sesame ya estaría realizada.

5.3.2. Utilización de Sesame

Para comenzar a trabajar con Sesame, lo primero que se debe hacer es encender el servidor. Para esto se debe acceder al directorio C:\apache-tomcat-6.0.41\bin y abrir el fichero startup.bat (Figura 11).

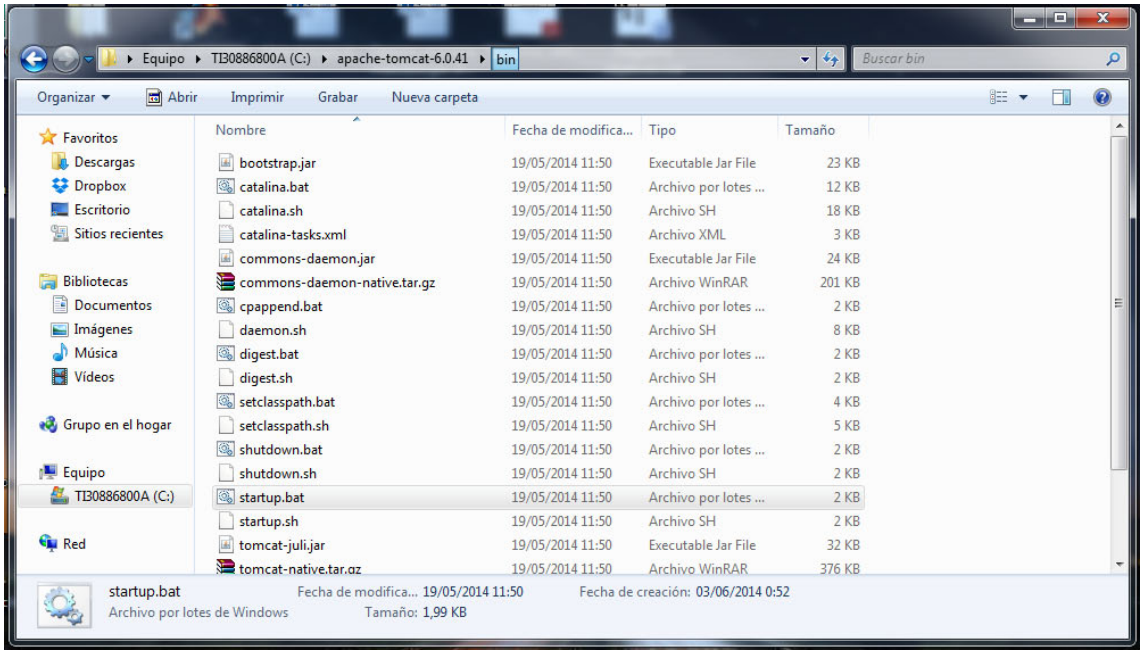


Figura 11. Archivo para arrancar el servidor propio

Al hacerlo aparecerá una ventana que mostrará cómo se han desplegado los archivos que existen en el servidor y el tiempo que ha tardado (Figura 12):

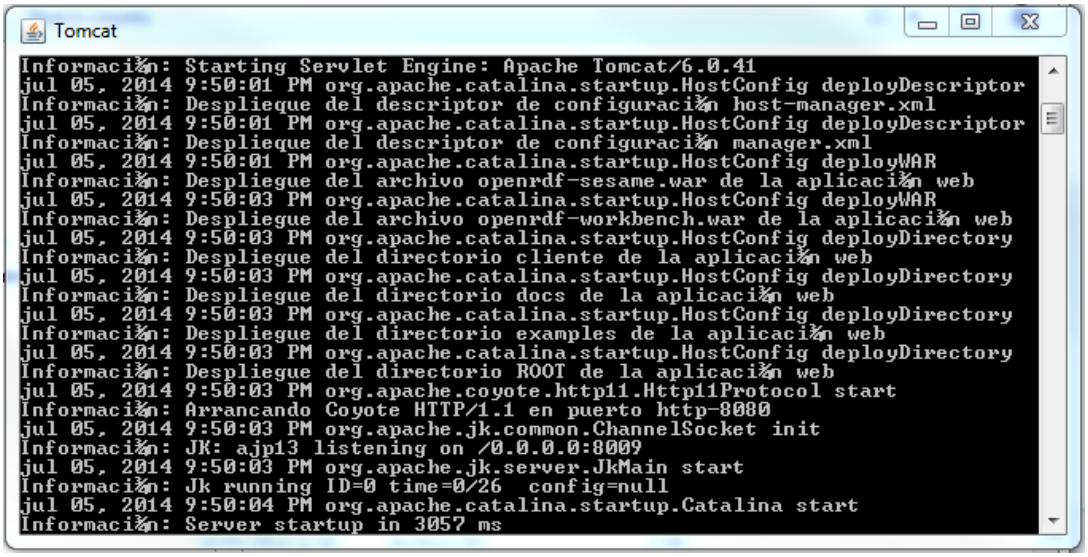


Figura 12. Ventana de control del servidor Tomcat

Una vez inicializado el servidor, se puede acceder a los archivos de Sesame instalados en el servidor introduciendo las siguientes URL en un navegador web:

- Acceso a la API de consulta: http://localhost:8080/openrdf-sesame
- Acceso a la interfaz de Sesame: http://localhost:8080/openrdf-workbench

La primera de ellas será utilizada posteriormente, para consultar a la plataforma desde una aplicación web, lo cual se tratará en el apartado 5.3.4. La segunda permite el acceso a la interfaz de tratamiento de datos RDF. Esta interfaz aparece reflejada en la Figura 13.

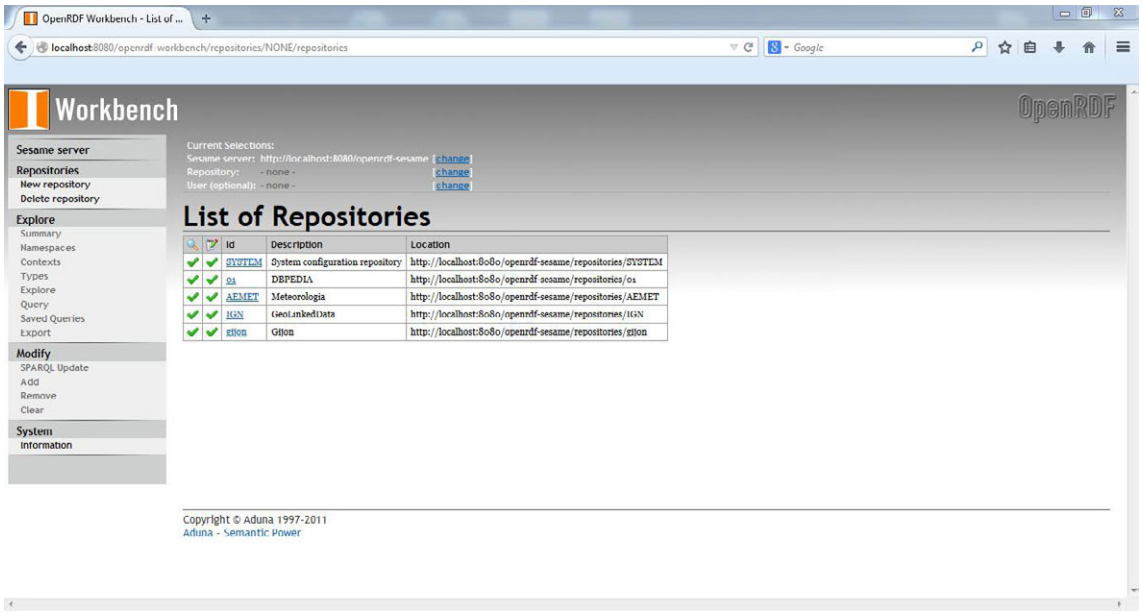


Figura 13. Interfaz del *software* Sesame

5.3.3. Posibilidades de Sesame

Dentro de la interfaz del *software* se encuentran diferentes menús que permiten el tratamiento de datos enlazados.

El primero de ellos es "Repositories". En él se ofrece la posibilidad de crear o eliminar un repositorio de datos enlazados. Al hacer clic en "New Repository" aparece la ventana de la Figura 14:

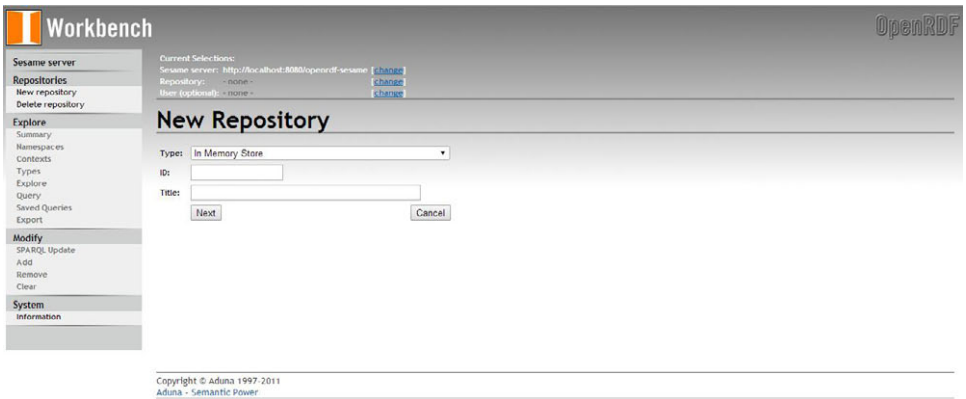


Figura 14. Creación de un nuevo repositorio en Sesame

En ella se debe seleccionar un nombre para el repositorio, un ID, que será el que se use para realizar la consulta a través de la petición HTTP y el tipo de repositorio a crear.

Los tipos de repositorios que permite crear aparecen en la Figura 15:

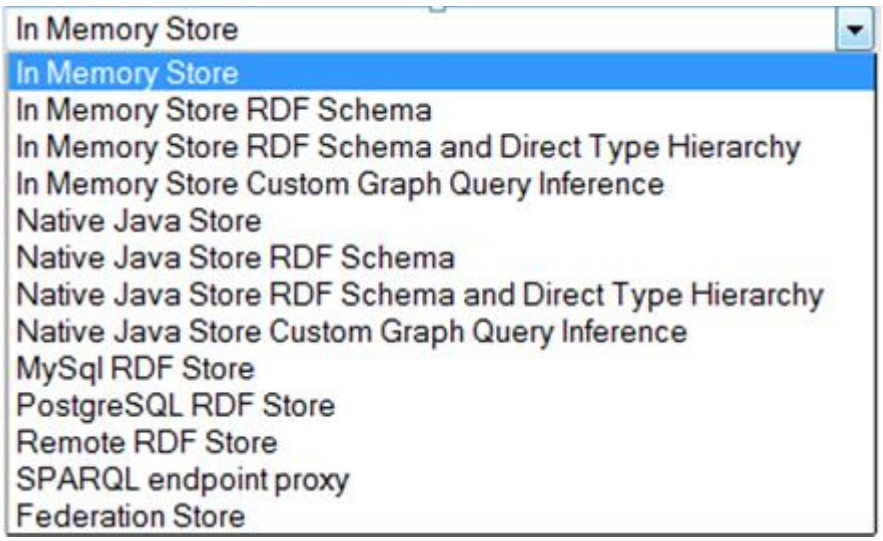


Figura 15. Tipos de repositorios disponibles en Sesame

Existe la opción de crear repositorios en un almacén en memoria, en almacén Java nativo, en una base de datos MySql RDF, en PostgreSQL RDF, desde un *endpoint* SPARQL externo y una federación de almacenes de datos. En el presente estudio se utilizarán los repositorios en memoria propia y en *endpoint* SPARQL.

Si se avanza en el menú seleccionando la opción de crear un repositorio en memoria el programa pide que se seleccione si el usuario prefiere un repositorio con almacenamiento persistente y el retraso máximo que quiere en la sincronización con la base de datos. Sin embargo, si se selecciona la opción de crear el repositorio a partir de un *endpoint* SPARQL, tan solo pide introducir la URL del *endpoint* correspondiente.

Para lograr que se activen los siguientes menús se debe seleccionar un repositorio de los que aparecen en el centro de la pantalla.

En el primero de los menús, "*Explore*", el programa permite realizar una exploración de los datos de cada repositorio. Este menú contiene las opciones que se muestran en la Figura 16.

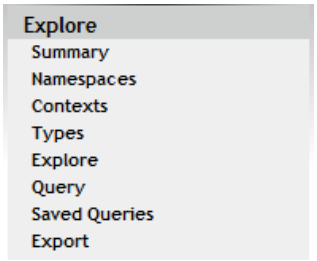


Figura 16. Menu *Explore* de Sesame

Summary muestra un resumen de la información del repositorio que esté seleccionado. Aparecen datos como el nombre y el ID del repositorio, el servidor al que pertenece y la localización de dentro del mismo, el número de sentencias que contiene [12] y el número de contextos etiquetados.

Namespaces muestra el número de espacios de nombres definidos en el repositorio. Permite tanto consultarlos como actualizar alguno nuevo.

Types muestra todos los conceptos definidos en el repositorio.

Explore permite introducir el URI de un concepto albergado en el repositorio seleccionado y devuelve el sujeto, predicado, objeto y contexto correspondientes.

Query permite realizar una consulta SPARQ o SeRQL al repositorio que está seleccionado. Se pueden limitar el número de resultados por página. Al realizar la consulta, devuelve los datos en forma de tabla, pero se puede seleccionar posteriormente el formato en el que se quieren obtener los datos (Binario, XML, JSON, TSV o CSV).

Saved Queries permite recuperar las consultas que se hayan guardado en el menú Query.

Export permite exportar los datos del repositorio seleccionado al formato que se seleccione. Los formatos permitidos aparecen en la Figura 17.

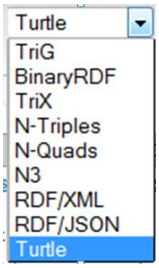


Figura 17. Formatos de exportación de datos de Sesame

En el segundo de los menús, llamado “Modify”, se modifican los datos del repositorio seleccionado. Las pestañas que contiene este menú aparecen reflejadas en la Figura 18.

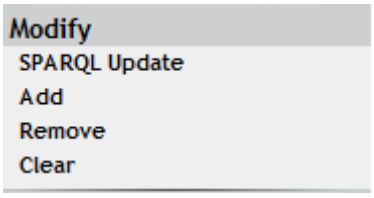


Figura 18. Menú *Modify* de Sesame

En *SPARQL Update* permite modificar los datos enlazados del repositorio seleccionado a través del lenguaje de actualización SPARQL 1.1 Update.

[12] Si el repositorio está enlazado a un repositorio externo muy extenso no es capaz de conseguir el número total de sentencias ni de contextos etiquetados.

En Remove permite eliminar alguna sentencia presente en el repositorio seleccionado.

En *Clear* permite eliminar un contexto completo (Un archivo RDF añadido anteriormente en nuestro caso)

En *Add*, permite añadir datos a un repositorio propio. Al hacer clic en esta pestaña se abre el menú representado en la Figura 19.

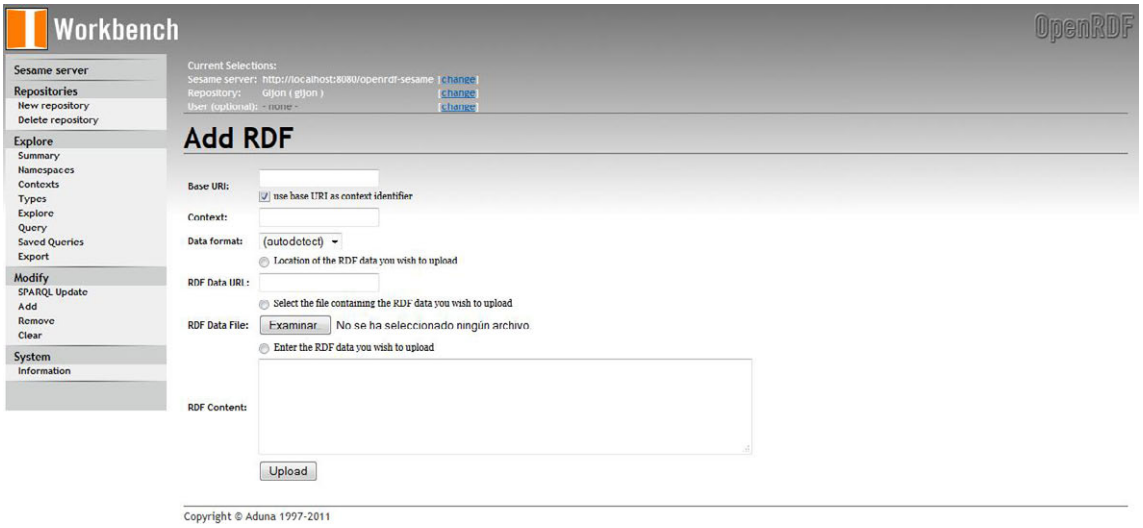


Figura 19. Pestaña *Add* de Sesame

En él permite añadir información en formato RDF de forma directa (*copy-paste*) en el cuadro RDF Content. Además de eso, permite hacerlo añadiendo el directorio en el que se encuentre un archivo RDF e incluso examinando el directorio.

Por último existe un último menú llamado *Information*, que ofrece información del sistema utilizado por el servidor. Se muestra información sobre la aplicación, sobre el servidor, y sobre la memoria utilizada.

5.3.4. Explotación externa de los datos a través de Sesame

La información almacenada en los repositorios, además de poder ser consultada en el citado menú *query*, puede ser consultada de forma externa a esta interfaz a través de peticiones HTTP y gracias a su API openrdf-sesame.war.

En el presente estudio lo anterior se realizará con la ayuda del lenguaje JavaScript y de JQUERY, una biblioteca de este lenguaje que aumenta la funcionalidad de este.

La idea es crear una cadena de caracteres que contenga, primeramente el acceso al repositorio al que se quiere acceder dentro de nuestro servidor. El comienzo de la cadena sería por tanto:

"http://localhost:8080/openrdf-sesame/repositories/repositorio_a_consultar"

Seguidamente se debe introducir la consulta en lenguaje SPARQL que se quiere realizar precedido de "?query=":

"?query=consulta_a_realizar

Y finalmente se añadiría el formato en el que se quieren obtener los datos. Se realizaría del siguiente modo:

"&Accept=application%2Fsparql-results%2Bjson&callback=?&format=application"

Enviando todo lo anterior como una cadena de caracteres con una petición GET, el servidor privado propio sobre el que se montó la aplicación Sesame, es capaz de devolvernos un fichero JSON con la información requerida en la consulta.

Esa información será procesada por una función posterior que tratará el JSON para representar finalmente la información en un visualizador de información geográfica.

La idea que se acaba de ilustrar aparece más ampliamente ilustrada en los siguientes apartados, donde se expone como se ha aplicado en ejemplos reales.

5.3.5. Datos de meteorología (AEMET).

El Ontology Engineering Group a través de su iniciativa AEMETLinkedData, ha transformado los datos de la Agencia Estatal de Meteorología a datos enlazados y abiertos. Con lo cual, del mismo modo que en Wikipedia, cada concepto está representado por una URI. Esta URI puede ser consultada a través de una *Wiki* asociada donde se muestran en forma de tabla todos los conceptos relacionados con un determinado concepto. Como ejemplo se muestra a continuación el URI que identifica la estación meteorológica de AEMET de A Coruña y su wiki correspondiente (Figura 20).

http://aemet.linkeddata.es/page/resource/WeatherStation/id08001

Figura 20. Wiki de AEMETLinkedData referente a la estación meteorológica de A Coruña

Además de obtener la información con la *Wiki* del concepto, la iniciativa cuenta con un endpoint SPARQL[13] al que se le pueden consultar datos meteorológicos de las distintas estaciones del estado español.

Aunque se intentó realizar la consulta de forma directa al *endpoint* sufriendo diversos problemas para solucionar la petición HTTP, se prefirió trabajar desde la herramienta Sesame.

[13] http://aemet.linkeddata.es/sparql

Se creó un repositorio de tipo "*SPARQL endpoint proxy*" nuevo cuyo ID era AEMET y su nombre también. De esta forma se puede utilizar como *endpoint* la siguiente raíz:

http://localhost:8080/openrdf-sesame/repositories/AEMET

Con esto se lograron evitar todos los problemas de Cross Domain que aparecían a la hora de enviar las peticiones directamente al *endpoint* del proyecto AEMETLinkedData.

5.3.6. Consultas realizadas a AEMETLinkedData y resultados

Las consultas SPARQL que se realizaron para consultar este repositorio son las siguientes:

- Obtener la latitud, longitud, nombre y código de todas las estaciones meteorológicas de AEMET. La consulta tiene la siguiente forma:

```
PREFIX aemetonto:    <http://aemet.linkeddata.es/ontology/>
PREFIX geo:    <http://www.w3.org/2003/01/geo/wgs84_pos#>
SELECT ?nombre ?lat ?lon ?codigo
WHERE {
?estacion rdf:type aemetonto:WeatherStation.
?estacion aemetonto:stationName ?nombre.
?estacion aemetonto:indsinop ?codigo.
?estacion geo:location ?loc.
?loc geo:lat ?lat.
?loc geo:long ?lon. }
```

- Obtener el nombre, el tipo de variable climática representada, y el valor de esa variable, limitando la consulta a un cierto número de resultados.

```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX xsd:    <http://www.w3.org/2001/XMLSchema#>
PREFIX ssn:    <http://purl.oclc.org/NET/ssnx/ssn#>
PREFIX aemet:          <http://aemet.linkeddata.es/ontology/>
PREFIX w3ctime:<http://www.w3.org/2006/time#>
PREFIX geo:    <http://www.w3.org/2003/01/geo/wgs84_pos#>
SELECT  ?nombre ?lat ?lon ?nameprop ?valueprop
WHERE {
?estacion aemet:indsinop "08001".
?estacion aemet:stationName ?nombre.
?estacion geo:location ?loc.
?loc geo:lat ?lat.
?loc geo:long ?lon.
?obs ssn:observedBy ?station .
?obs ssn:observedProperty ?prop.
?prop rdfs:label ?nameprop.
?obs aemet:valueOfObservedData ?valueprop .
FILTER( ?prop =
<http://aemet.linkeddata.es/resource/TemperatureAmbientProperty/TA>) } LIMIT 10
```

En este caso se obtienen los 10 últimos valores de temperatura ambiente registrados en la estación meteorológica de la Coruña.

5.3.7. Explicación del proceso

Una vez almacenado en una variable un conjunto de caracteres que represente alguna de las consultas anteriores, se invoca una función, a la que se le pasará el valor de esta variable. Esta función será capaz de concatenar la variable con la consulta que se le pasó al invocarla, con la raíz del *endpoint* correspondiente al repositorio de Sesame y el formato que se quiere obtener. Todo esto lo hará dentro de una función de JQUERY que enviará una petición GET y almacenará en una variable los datos en JSON.
Una vez hecho esto se enviara la citada variable a otra función que procese el JSON y permita representarlo en el visualizador.

El código que realiza el citado procedimiento es el siguiente:

```
function sparqlQueryJsonmeteo(queryStr) {
$.get("http://localhost:8080/openrdf-
sesame/repositories/AEMET?query="+queryStr+"&Accept=application%2Fsparql-
results%2Bjson&callback=?&format=application",{}, function(data) {

myCallbackest(data);//Función que va a procesar el JSON
});
```

5.3.8. Datos de GeoLinkedData (IGN)

Este proyecto también fue desarrollado por el Ontology Engineering Group y cuenta con información de diversas fuentes como el NGCE (Nomenclátor Geográfico Conciso de España), el NOMGEO (Base de datos de nombres geográficos georreferenciados del IGN), la BTN25 (Base Topográfica Nacional a escala 1:25.000) y la BCN 200 (Base Cartográfica Nacional a escala 1:200.000.

Al igual que con los datos de AEMET y los de DBpedia, cada concepto tiene asociado una URI, la cual conduce a una wiki en el que aparece información referente al mismo.

La URI asociada al aeropuerto de Alvedro en A Coruña sería así:

http://geo.linkeddata.es/page/resource/Aeropuerto/Coru%C3%B1a%2C%20Aeropuerto%20de%20A%20

La *Wiki* asociada aparece en la Figura 21:

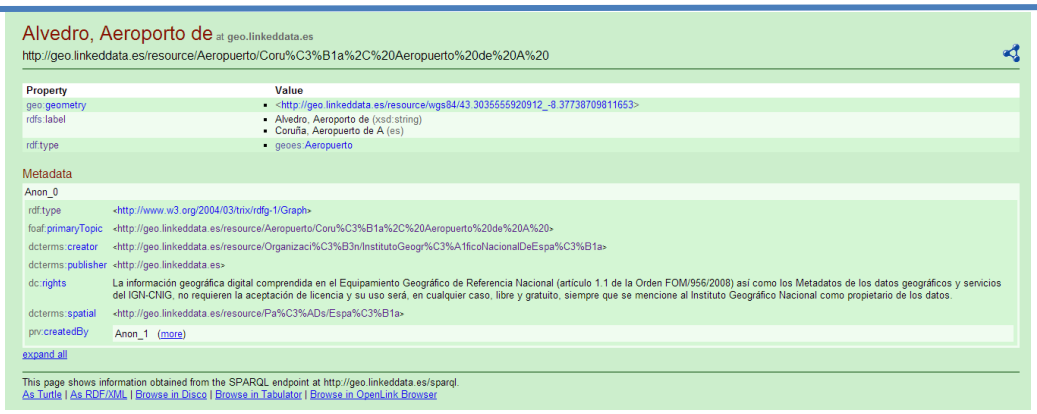


Figura 21. *Wiki* de GeoLinkedData referente al aeropuerto de Alvedro

Al igual que en los casos anteriores, existe un *endpoint* SPARQL de la iniciativa GEOLinkedData [14]que permite obtener los datos en RDF a través de peticiones HTTP.

Del mismo modo que sucedía con los datos enlazados de AEMET, se han producido diversos problemas al enviar la petición HTTP. Estos problemas se han solucionado definiendo un repositorio en Sesame que consulte los datos de este *endpoint*.

5.3.9. Consultas realizadas a GeoLinkedData y resultados

Las consultas SPARQL que se realizaron para consultar este repositorio son las siguientes:

- Nombre, latitud y longitud de algunos elementos de hidrografía existentes en el proyecto:

```
PREFIX geo: <http://www.w3.org/2003/01/geo/wgs84_pos%23>
PREFIX geoes: <http://geo.linkeddata.es/ontology/>
SELECT  ?nombre ?lat ?lon {
?concepto rdf:type geoes:Nacimiento.
?concepto rdfs:label ?nombre.
?concepto geo:geometry ?loc.
?loc geo:lat ?lat.
?loc geo:long ?lon.) }
```

En el caso concreto de la consulta anterior se obtendrían todos los nacimientos de ríos del territorio español almacenados en el repositorio. Para obtener otros datos de hidrografía se debe sustituir la palabra Nacimiento por la palabra Piscina, Playa, Laguna, Pozo, Pantano, Mar, Marisma o Faro en la etiqueta geoes:Nacimiento, en función de la información que se quiera obtener.

Uno de los mayores inconvenientes de esta iniciativa es que la cantidad de información almacenada para cada concepto es muy limitada, lo cual no permite observar la potencia real de los datos enlazados.

[14] http://geo.linkeddata.es/sparql

5.3.10. Explicación del proceso

De la misma forma que en la explicación del proceso para la obtención de los datos en JSON del repositorio de AEMET, se invoca una función, a la que se le pasará el valor de una variable que almacenará la consulta a realizar. Esta función será capaz de concatenar la variable con la consulta que se le pasó al invocarla, con la raíz del *endpoint* correspondiente al repositorio de Sesame y el formato que se quiere obtener. Todo esto lo hará dentro de una función de JQUERY que enviará una petición GET y almacenará en una variable los datos en JSON.
Una vez hecho esto se enviará la citada variable a otra función que procese el JSON y permita representarlo en el visualizador.

El código que realiza el citado procedimiento es el siguiente:

```
function sparqlQueryJsongeolinked (queryStr) {
$.get("http://localhost:8080/openrdf-
sesame/repositories/IGN?query="+queryStr+"&Accept=application%2Fsparql-
results%2Bjson&callback=?&format=application",{}, function(data) {

myCallbackgeolinked(data);//Función que va a procesar el JSON
})
```

5.4. Extracción de datos de un repositorio privado con la ayuda de Sesame

Como se mencionó anteriormente, Sesame es capaz de trabajar con datos almacenados en un repositorio personal. Esto se ha aplicado con los datos RDF que proporciona el Ayuntamiento de Gijón. Aunque dispone de *endpoint* SPARQL para extraer información enlazada, este no está operativo en este momento. Debido a esto se ha decido realizar una descarga directa de información en formato RDF de la página web, que contuviera información geográfica, almacenándola en una carpeta propia.

El proceso de creación de un repositorio con datos propios en Sesame aparece explicado en el apartado 5.4.1.

5.4.1. Datos Gijón (Ayuntamiento de Gijón)

Como ya se mencionó anteriormente, el Ayuntamiento de Gijón cuenta con un gran repositorio de datos enlazados en su página web, así como con un *endpoint* SPARQL. Debido a la inoperatividad del mismo, se ha decidido trabajar con algunos de los datos disponibles en un repositorio privado.

Para ello, lo primero que se hizo fue una descarga de ficheros RDF con información relativa a bancos, aparcamientos, pubs, bancos, bibliotecas, boleras, camping, campos de futbol, cines y comercios. Estos ficheros se almacenaron en una misma carpeta dentro de la carpeta del servidor.

Para crear un nuevo repositorio se ha acudido a la interfaz de Sesame y se ha pulsado en “New repository”. El tipo de repositorio que se ha creado es ”In Memory Store” con ID “gijon” y nombre Gijón.

Una vez creado el repositorio, se ha marcado este repositorio por defecto y se ha acudido al botón “add”. En él se permite añadir un archivo de datos enlazándolos al repositorio. Para hacerlo correctamente[15] se debe escribir el directorio dentro del servidor privado donde están almacenados datos en el input que se llamen “RDF Data URL:”. Un ejemplo de esto aparece en la Figura 21.

http://localhost:8080/openrdf-sesame/DatosGijon/aparcamientos.rdf

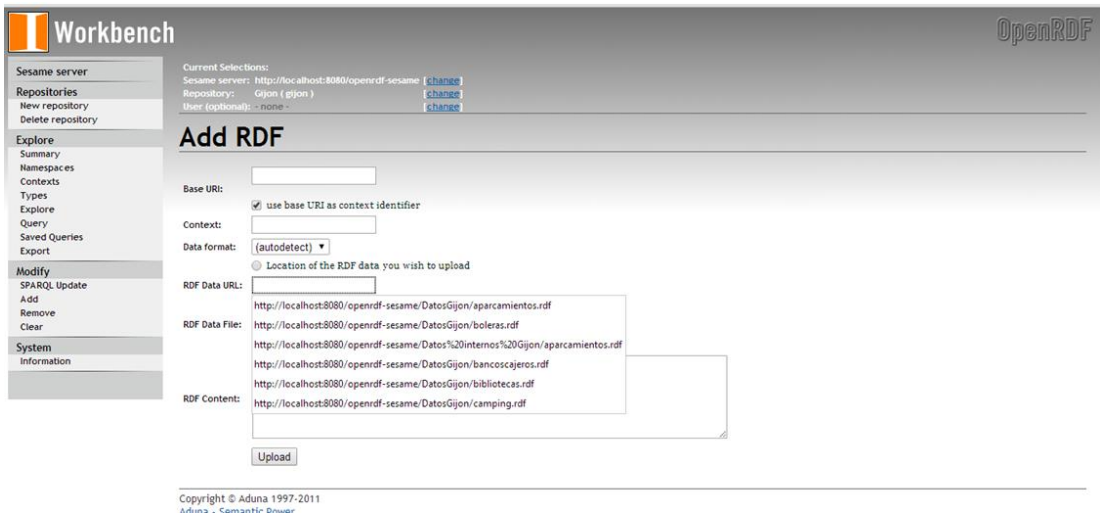


Figura 22. Creación de un repositorio privado en Sesame

Esto se hace con todos los datos almacenados en el citado directorio para crear el repositorio de datos propios.

A diferencia de todos los datos contemplados hasta el momento, los que proporciona el Ayuntamiento de Gijón no tiene los conceptos almacenados en URIs, sino que simplemente aparecen los datos en formato RDF. Un ejemplo de esto sería el fragmento de datos en RDF de los aparcamientos de Gijón que aparece en la Figura 23.

[15] Otra forma de introducir la información de forma más cómoda sería examinando el directorio en el botón “examinar”, pero al hacerlo así, introduce la URI del contexto como /fakepath, los cual dificulta su posterior utilización en las consultas SPARQL.

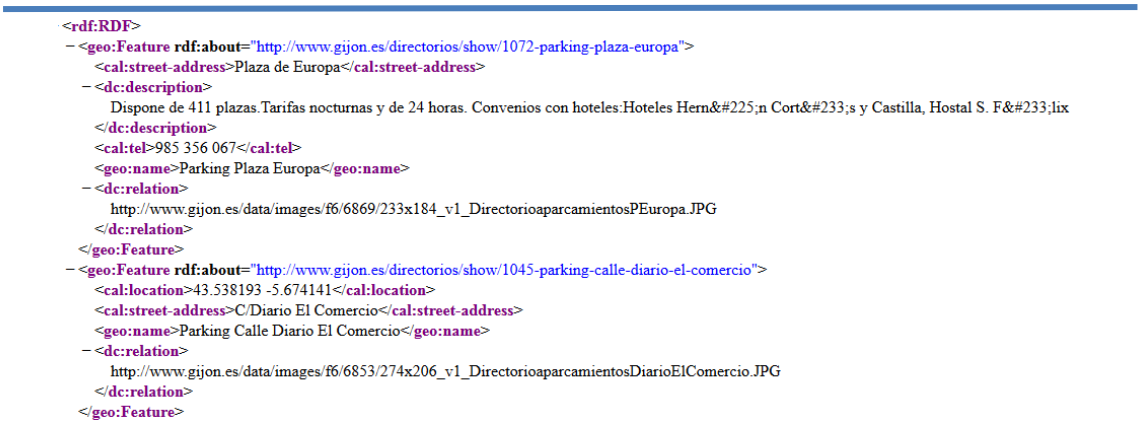


Figura 23. Ejemplo de datos en RDF del Ayuntamiento de Gijón

5.4.2. Consultas realizadas al repositorio propio y resultados

Las consultas SPARQL realizadas al repositorio de datos propio son las siguientes:

- Obtención del nombre y localización de los elementos almacenados en un determinado contexto o repositorio.

```
PREFIX geo: <http://www.geonames.org/ontology#>
PREFIX cal: <http://www.w3.org/2002/12/cal#>
PREFIX dc: <http://purl.org/dc/elements/1.1#>
SELECT ?nombre ?localizacion
FROM <http://localhost:8080/openrdf-sesame/DatosGijon/aparcamientos.rdf>
WHERE{
?datasetURI geo:name ?nombre.
?datasetURI cal:location ?localizacion. }
```

En el FROM se especifica el contexto del que se va extraer información, en "?localización" se almacena un texto que contiene la latitud y longitud y deberá ser procesado posteriormente para separar los dos valores.

Para extraer información de otros ficheros se debe substituir la palabra "aparcamientos" por la que identifique a cada uno de ellos, por ejemplo, "boleras", "camposdefutbol" o "comercios".

5.4.3. Explicación del proceso

El proceso para obtener la información en JSON es el mismo que con los dos repositorios anteriores, ya que se vuelve a enviar la petición al servidor propio. Se vuelve a crear una función a la que se le pasa una variable que almacena la consulta a realizar. Se concatena con la raíz del servidor propio y el formato de recepción y se envía con una petición GET.

La parte del código que realiza esto es:

```
function sparqlQueryJsongijon(queryStr) {
alert(servicio);
        $.get("http://localhost:8080/openrdf-sesame/repositories/gijon?query="+queryStr+"&Accept=application%2Fsparql-results%2Bjson&callback=?&format=application",{}, function(data) {
myCallbackgijon(data);// A esta función se le pasa el JSON obtenido de la petición. });}
```

6. Desarrollo de la aplicación

Con el objetivo de poner de manifiesto el funcionamiento de las anteriores consultas de información geográfica enlazada, se ha implementado una aplicación en lenguaje JavaScript, que además de obtener los datos a través del lenguaje SPARQL, tiene otras funcionalidades que la harán más interactiva con el usuario.

Esta aplicación tratará de cumplir con los puntos 4, 5 y 6 que se marcaron en el apartado de Objetivos (Apartado 3).

Será capaz por tanto de representar en un visualizador de información geográfica los datos geográficos extraídos del RDF o de un *endpoint* SPARQL externo.

Se diseñará una interfaz que permita seleccionar al usuario que información desea obtener, abstrayendo totalmente todo el procedimiento que se encuentra detrás de la aplicación.

Se incluirá así mismo, un módulo capaz de fijar un marco de trabajo determinado que permita restringir el ámbito geográfico de afectación de cada una de las consultas.

El aspecto que tendrá la aplicación final aparece en la Figura 24.

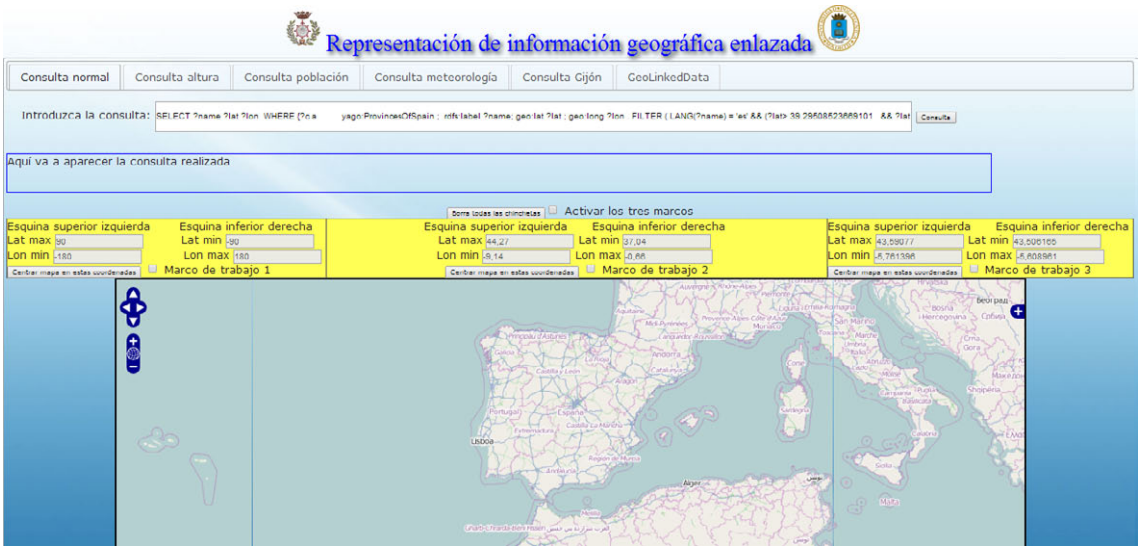


Figura 24. Interfaz final de la aplicación

En la parte superior de la interfaz aparecen las pestañas para moverse a través de las diferentes funciones de la aplicación [23]. Justo debajo aparecen las opciones para introducir datos de cada una de las pestañas de la aplicación. Más abajo aparece un recuadro azul en el van a aparecer todas las consultas SPARQL que el programa realiza internamente.

Si se continúa descendiendo se pueden observar 3 ventanas de color amarillo, este es el módulo para definir el marco de trabajo, sobre el cuál se hablará más extensamente en los próximos apartados.

Por último en la parte inferior de la aplicación aparece un mapa interactivo de Open Street Map, sobre el que se podrá visualizar la información.

En la Figura 25 se muestra un diagrama que pone de manifiesto el funcionamiento de la aplicación.

6.1.Flujo de trabajo de la aplicación

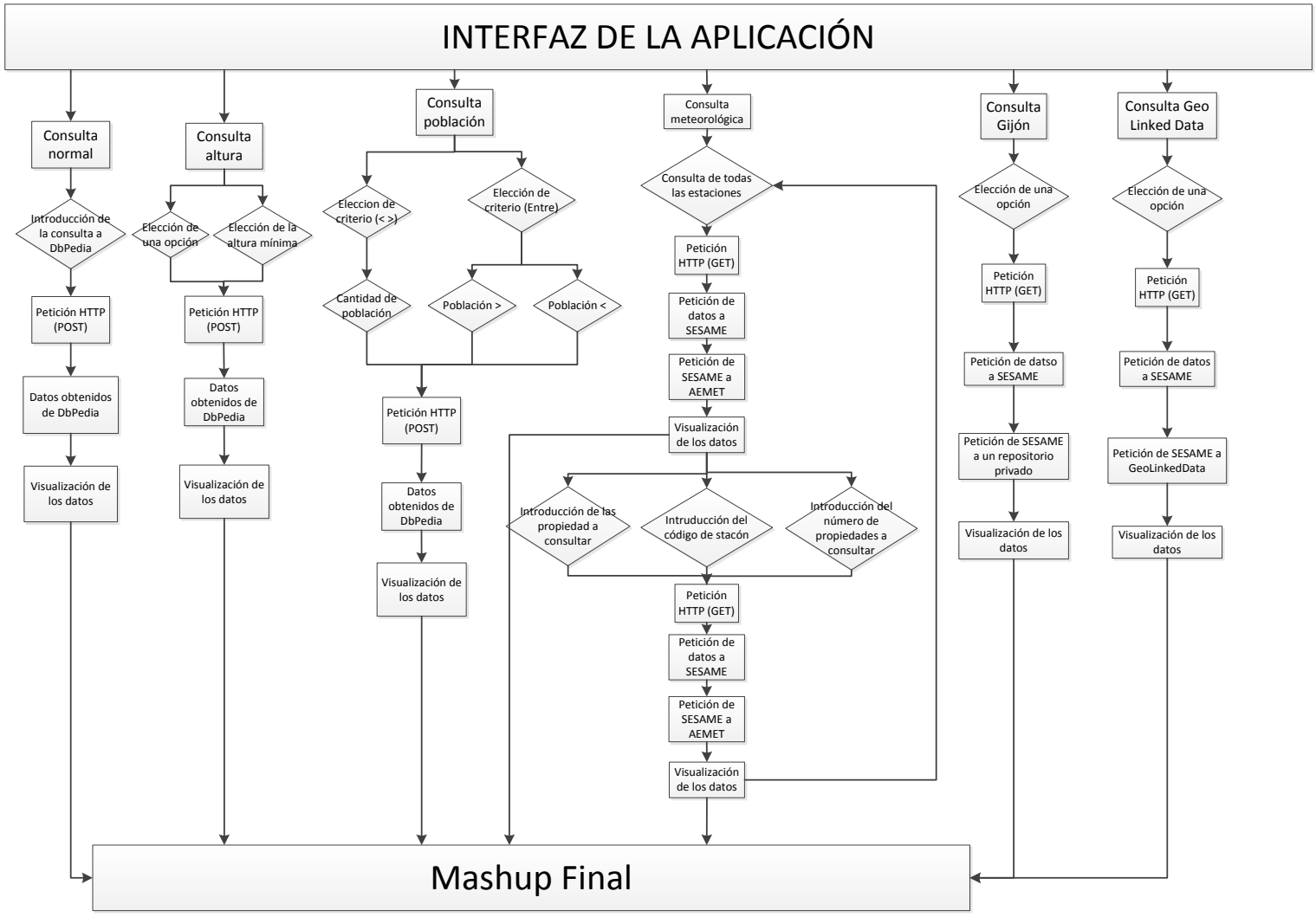


Figura 25. Flujo de trabajo de la aplicación

6.2. Funcionalidades

En este apartado se van a explicar en profundidad todas las funcionalidades que incorpora la aplicación, examinando las partes más importantes de su código, y exponiendo su funcionamiento completamente.

6.2.1. Menú de marcos de trabajo

El objetivo de este menú es restringir interactivamente uno o varios marcos de trabajo que limiten la extensión de las consultas realizadas. El aspecto de la interfaz se muestra en la Figura 26.

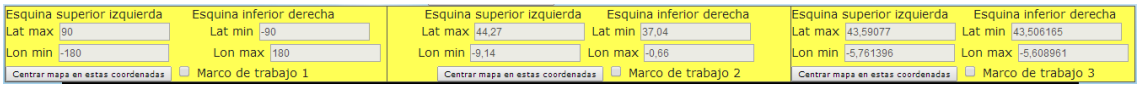


Figura 26. Módulo de marcos de trabajo

Cada uno de los rectángulos amarillos se corresponde con un marco de trabajo. En ellos se pueden observar 4 cuadros de texto en cada uno, en los cuales se pueden introducir manualmente la latitud y longitud de las esquinas del recuadro que se quieren definir como marco de trabajo.

En el botón "Centrar mapa en estas coordenadas" se centra el mapa en las coordenadas que estén escritas en el *textbox* correspondiente, esté activado o no.

El código para lograr esto aparece en la Figura 27

```
// Función para centrar
function centrar(){
var lonmin = document.getElementById('lonmin').value
    var latmin = document.getElementById('latmin').value
    var lonmax = document.getElementById('lonmax').value
    var latmax = document.getElementById('latmax').value
    map.zoomToExtent(new OpenLayers.Bounds(lonmin, latmin, lonmax,
    latmax).transform(new OpenLayers.Projection("EPSG:4326"),map.getProjectionObject()));
}
```

Figura 27. Código JavaScript para centrar el mapa en un rectángulo

Para activar la escritura en los cuadros de texto y la funcionalidad de los marcos, se debe pulsar el *checkbox* correspondiente al marco que se desea utilizar. Esto se logra con el código de la Figura 27.

```
    jQuery(document).ready(function () {
$("#myCheck1").click(function () {
  $('#latmin').attr("disabled", !$(this).is(":checked"));
  $('#latmax').attr("disabled", !$(this).is(":checked"));
  $('#lonmin').attr("disabled", !$(this).is(":checked"));
  $('#lonmax').attr("disabled", !$(this).is(":checked"));
 if (!$(this).is(":checked")){
   var primerclic=true;
  } else{
   darcoord();
   }
});
```

Figura 28. Código JavaScript para activar los cuadros de texto

En este caso, cuando el *checkbox* está desactivado, los cuadros de texto también se desactivan, en caso contrario entra en la función darcoord().

Como se ha mencionado, en el momento en el que alguno de los *checkbox* estén activados, se lama la función darcoord() que habilita la opción de capturar la latitud y longitud de un punto haciendo clic en el mapa. Esto se consigue gracias a la librería de OpenLayers.

El código para situar las coordenadas en el primer marco de trabajo aparece en la Figura 29 y en la Figura 30.

```
function darcoord(){
        var primerclic=true;
        var lonlat1=0;
        var lonlat2=0;
        var latinicial;
        var longinicial;
        var latfinal;
        var longfinal;

OpenLayers.Control.Click = OpenLayers.Class(OpenLayers.Control, {
            defaultHandlerOptions: {
                'single': true,
                'double': false,
                'pixelTolerance': 0,
                'stopSingle': false,
                'stopDouble': false
            },
            initialize: function(options) {
                this.handlerOptions = OpenLayers.Util.extend(
                    {}, this.defaultHandlerOptions
                );
                OpenLayers.Control.prototype.initialize.apply(
                    this, arguments
                );
                this.handler = new OpenLayers.Handler.Click(
                    this, {
                        'click': this.trigger
                    }, this.handlerOptions
                );
            },
            trigger: function(coord) {
            if ($("#myCheck1").is(":checked")){
                if (primerclic==true){
                    lonlat1 = map.getLonLatFromPixel(coord.xy).transform('EPSG:3857' , 'EPSG:4326' );
```

Figura 29. Código JavaScript para realizar el clic interactivo (Parte 1)

```
                primerclic=false;
                latinicial= lonlat1.lat;
                longinicial= lonlat1.lon;
                document.getElementById('latmax').value = latinicial;
                document.getElementById('lonmin').value = longinicial;
            }   else {
                lonlat2= map.getLonLatFromPixel(coord.xy).transform('EPSG:3857' , 'EPSG:4326' );
                latfinal= lonlat2.lat;
                longfinal= lonlat2.lon;
                document.getElementById('latmin').value = latfinal;
                document.getElementById('lonmax').value = longfinal;
                primerclic=true;
                $('#latmin').attr("disabled", true);
                $('#latmax').attr("disabled", true);
                $('#lonmin').attr("disabled", true);
                $('#lonmax').attr("disabled", true);
                document.getElementById("myCheck1").checked=false;
            }
            }
        }
    });
        var click = new OpenLayers.Control.Click();
        map.addControl(click);
        click.activate();
}
```

Figura 30. Código JavaScript para realizar el clic interactivo (Parte 2)

Como se puede observar primero se declaran unas cuantas variables. La primera de ella (primerclic) es un booleano, es decir, almacena valores falso o verdadero. Más adelante define una función de la librería de OpenLayers que controla el clic del ratón. Dentro de esta función se definen primero una serie de opciones para habilitar el clic simple o el doble, se inicializa la función y lanza un *trigger* que será el que obtenga las coordenadas.

Como se puede ver, en caso de que el *checkbox* esté activado, y de que "primerclic" sea cierto, entra en la primera sentencia del *if*, la cual escribe la latitud y la longitud sus cuadros de texto correspondientes y coloca el "primerclic" como false. Gracias a esto, al hacer el segundo clic en el mapa, entra en la segunda sentencia del condicional y almacena unas segundas coordenadas, dibujándolas en los cuadros de texto correspondientes. De esta forma quedan escritas las coordenadas en el marco de referencia.

Este código debe copiarse para cada uno de los marcos de referencia para así hacer operativos a los tres marcos.

Una vez que se tienen definidos los marcos a utilizar es importante tener en cuenta que las consultas sólo se harán operativas cuando alguno de los *checkbox* esté activado, de no ser así, la aplicación enviará una advertencia.

El siguiente paso es introducir en cada una de las consultas el valor de la latitud y la longitud de cada *textbox*. Esto se analizará en el apartado 5.3.2.

Además de los marcos en sí, existe un botón que permite eliminar todas las chinchetas del mapa y un *checkbox*, que al activarlo, se activan los tres marcos al mismo tiempo (Figura 31).

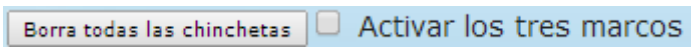


Figura 31. Botón para borrar todos los marcadores y *Checkbox* para activar todos los marcos de trabajo

El primero de ellos llama a una función que elimina la capa en la que se almacenan las chinchetas y crea una nueva completamente vacía. Se logra con el código de la Figura 32:

```
function borralayers(){

lyrMarkers.displayInLayerSwitcher = false; .
    lyrMarkers.setVisibility();
    lyrMarkers = new OpenLayers.Layer.Markers("Markers");
    map.addLayer(lyrMarkers);
}
```

Figura 32. Código JavaScript para borrar los marcadores

El segundo se ayuda de jQuery para activar o desactivar los marcos. Esto se logra con código de la Figura 33:

```
$("#myChecktotal").click(function () {
 $('#latmin').attr("disabled", !$(this).is(":checked"));
   $('#latmax').attr("disabled", !$(this).is(":checked"));
   $('#lonmin').attr("disabled", !$(this).is(":checked"));
   $('#lonmax').attr("disabled", !$(this).is(":checked"));
 $('#latmin2').attr("disabled", !$(this).is(":checked"));
   $('#latmax2').attr("disabled", !$(this).is(":checked"));
   $('#lonmin2').attr("disabled", !$(this).is(":checked"));
   $('#lonmax2').attr("disabled", !$(this).is(":checked"));
 $('#latmin3').attr("disabled", !$(this).is(":checked"));
   $('#latmax3').attr("disabled", !$(this).is(":checked"));
   $('#lonmin3').attr("disabled", !$(this).is(":checked"));
   $('#lonmax3').attr("disabled", !$(this).is(":checked"));
   if($(this).is(":checked")){
   document.getElementById("myCheck1").checked=true;
    document.getElementById("myCheck2").checked=true;
     document.getElementById("myCheck3").checked=true;
     darcoord();
     darcoord2();
     darcoord3();
   } else{
   document.getElementById("myCheck1").checked=false;
    document.getElementById("myCheck2").checked=false;
     document.getElementById("myCheck3").checked=false;
   }
 });
```

Figura 33. Código JavaScript para activar o desactivar todos los marcos a la vez

De este modo se consigue realizar un módulo capaz de trabajar con tres marcos de trabajo simultáneos, mejorando la usabilidad de la aplicación y de los datos enlazados.

6.2.2. Consulta directa

El menú de consultas aparece desarrollado en la parte superior de la aplicación. Su interfaz está compuesta por un conjunto de pestañas interactivas, a través de las cuales se puede acceder a un menú o a otro. El aspecto de esta interfaz aparece representado en la Figura 34.

Figura 34. Pestañas interactivas de la aplicación

Su aspecto está dispuesto respecto al archivo CSS correspondiente y su funcionalidad implementada con el lenguaje JQUERY.

A continuación se detallan en mayor profundidad las funcionalidades de cada una de las pestañas.

La funcionalidad de la primera de las pestañas es la realización de cualquier consulta directa a DBpedia introducida manualmente en el cuadro de texto habilitado al efecto.

Debe tenerse en cuenta que los datos que debe devolver la consulta sean el nombre del elemento a representar, latitud y longitud. Para ello se deben situar después del SELECT de cualquier consulta, los términos, ?name, ?lat, ?lon, con esta disposición y sin añadir ninguno más.

La forma que tiene la interfaz al entrar en esta pestaña aparece en la Figura 35.

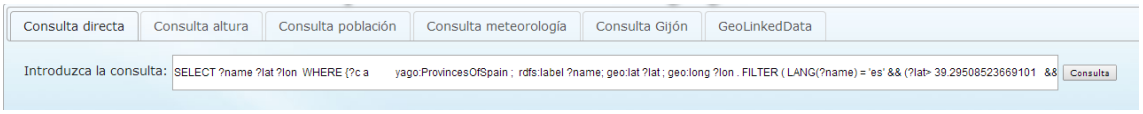


Figura 35. Interfaz de la consulta directa

En ella aparece el citado cuadro de texto con una consulta por defecto dibujada, y un botón que llama a la función que realiza la consulta directa.

En el caso de enviar la consulta sin activar ninguno de los marcos de trabajo, la aplicación enviara un mensaje (Figura 36) en el cual se invita a utilizarlo, pero realizará la consulta tal cual está escrita en el cuadro de texto.

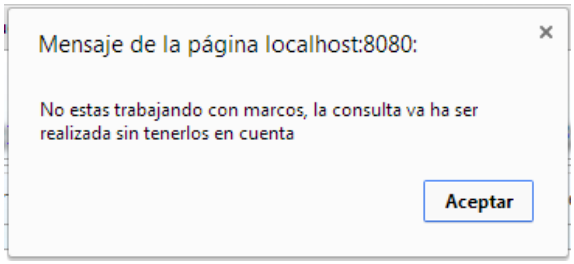


Figura 36. Mensaje de aviso al trabajar sin marcos

Esto se logra gracias a la siguiente parte de código, que es la que se ejecuta en cuanto se pulsa el botón Consulta (Figura 37).

```
function consulta(){

if (!$("#myCheck1").is(":checked") && !$("#myCheck2").is(":checked") && !$("#myCheck3").is(":checked")){
alert("No estas trabajando con marcos, la consulta va ha ser realizada sin tenerlos en cuenta");
var querycon= document.getElementById('consulta').value
document.getElementById("cuadroconsulta").innerHTML = querycon;
var endpoint = "http://dbpedia.org/sparql"
sparqlQueryJson(querycon, endpoint, myCallback, true)
return;
}
```

Figura 37. Código JavaScript que realiza la consulta directa si los marcos están desactivados

En él se comprueba que todos los checkbox están desactivados. Si es así, lanza el mensaje de alerta, almacena en una variable la consulta, la dibuja en el cuadro azul y envía la consulta y el *endpoint* a la función para realizar la petición HTTP. Esta función ya ha sido explicada en el apartado 5.2.

De permanecer marcado alguno de los marcos de trabajo, la función tiene en cuenta las coordenadas almacenadas en los mismos. Esto se logra gracias al código representado en la Figura 38.

```
if ($("#myCheck1").is(":checked"))
var latfinal= document.getElementById('latmin').value
var latinicial= document.getElementById('latmax').value
var longinicial= document.getElementById('lonmin').value
var longfinal= document.getElementById('lonmax').value
var posit = "FILTER ( LANG(?name) = 'es' &&  ?lat> "+latfinal+"  && ?lat < "+latinicial+" && ?lon> "+longinicial+"  && ?lon < "+longfinal+")}";
var query= document.getElementById('consulta').value

if (/FILTER/.test(query))
{
var n = query.search(/FILTER/i);
var resul = query.substr(0,n)
var queryfin = resul+posit

} else{
var n = query.search(/}/i);
var resul = query.substr(0, n)
var queryfin = resul+posit;
}

document.getElementById("cuadroconsulta").innerHTML = queryfin;
var endpoint = "http://dbpedia.org/sparql"
sparqlQueryJson(queryfin, endpoint, myCallback, true)
```

Figura 38. Código JavaScript que realiza la consulta directa si alguno de los marcos está activado

En el primero se comprueba si un determinado marco está activado. Si lo está, almacena en diversas variables las latitudes y longitudes correspondientes a la esquina superior izquierda y a la esquina inferior derecha definidas en el marco correspondiente. Con esas variables se crea una nueva que almacena la cadena de caracteres que contiene la parte de una consulta SPARQL correspondiente al FILTER. Seguidamente extrae la consulta introducida en el cuadro de texto y lo almacena en una variable.

Una consulta SPARQL puede o no contener un FILTER o filtro de los datos, eso es lo que se comprueba con el condicional inferior, si aparece FILTER, almacena en “resul” todo lo que va antes del filter y lo concatena con las nuevas coordenadas, para crear un string con el nuevo marco de trabajo definido. SI no aparece un filter, busca hasta la ultima llave de la consulta, coge todo lo anterior y lo concatena a la variable donde se almacenaba el marco de

trabajo. De este modo se contemplan los dos casos. Al finalizar el condicional actúa igual que en la primera parte, envía la consulta y el *endpoint* a la función que realizará la petición http.

El código anterior estará triplicado para cada uno de los marcos de trabajo de la aplicación.

El nombre de la función que envía la petición HTTP, lleva por nombre myCallback. Esta función es la encargada de procesar los datos en JSON que se reciben de la consulta SPARQL. El código de esta función se muestra en la Figura 39.

```
// Funcion que se llama al ejecutar correctamente la query SPARQL
//        e intenta añadir la información al mapa
function myCallback(str) {
    str = eval('(' + str + ')');
    for(var i = 0; i<  str.results.bindings.length; i++) {
        var la = str.results.bindings[i].lat.value;
        var lo = str.results.bindings[i].lon.value;
        var name = str.results.bindings[i].name.value;
        addMarker(lo,la,name);
    }
}
```

Figura 39. Código JavaScript para procesar el JSON de la consulta directa

En él, se transforma el JSON en un objeto reconocible por JavaScript a través de la función eval(). Tras esto se crea un bucle que va recorriendo todos los elementos del JSON, y extrae en las variables “la”, “lo” y “name”; la latitud, la longitud y el nombre de los datos que se están manejando.

Esos datos son enviados a otra función llamada “addMarker”, que será la encargada de dibujar las chinchetas que representaran los elementos extraidos de la consulta. La forma de esta función aparece reflejada en la Figura 40.

```
// Funcion para añadir un marcador y su creador de popup asociado
function addMarker(lng, lat, info) {
    var pt = new OpenLayers.LonLat(lng, lat).transform(new OpenLayers.Projection("EPSG:4326"),map.getProjectionObject());
    var feature = new OpenLayers.Feature(lyrMarkers, pt);
    feature.closeBox = true;
    feature.popupClass = popupClass;
    feature.data.popupContentHTML = info;
    feature.data.overflow = "auto";
    var servicio = document.getElementById('datogijon').value;
    var marker = new OpenLayers.Marker(pt, icon.clone());
    var markerClick = function(evt) {
        if (currentPopup != null && currentPopup.visible()) {
            currentPopup.hide();
        }
        if (this.popup == null) {
            this.popup = this.createPopup(this.closeBox);
            map.addPopup(this.popup);
            this.popup.show();
        }
        else {
            this.popup.toggle();
        }
            currentPopup = this.popup;
            OpenLayers.Event.stop(evt);
    };
    marker.events.register("mousedown", feature, markerClick);
    lyrMarkers.addMarker(marker);
}
```

Figura 40. Código JavaScript para dibujar marcadores en el mapa

Como se puede ver, la función procesa los datos de latitud y longitud y los almacena en una variable y después la utiliza para crear la variable “marker”, que será la utilizada para añadir las chinchetas con lyrMarkers.addMarker(marker);.

Además de esto también activa la posibilidad de mostrar un PopUp con la información almacenada en la variable “info”.

6.2.3. Consulta altura

En este menú se muestra de forma más cercana al usuario el resultado de consultar información enlazada en DBpedia. Es capaz de representar la localización de elementos como las capitales de provincia de Ecuador, las capitales de estado de Estados Unidos, las capitales de estado de Alemania, las entidades de población etiquetadas como ciudades en DBpedia, las capitales de Europa, los volcanes activos y las montañas, filtrándolas por su altura.

La forma que tiene la interfaz en el caso de esta pestaña aparece en la Figura 41.

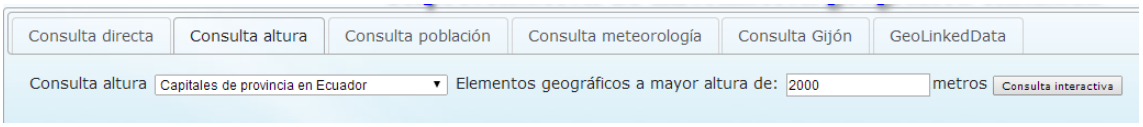


Figura 41. Interfaz de la pestaña Consulta altura

En ella se puede seleccionar de forma interactiva el elemento sobre el que se quiere obtener información y la altura mínima que debe tener. Es decir, la consulta seleccionará todos los elementos del tipo que se marque en el desplegable y que tengan una altura superior a la indicada en el cuadro de texto habilitado al efecto.

El código que consigue hacer esto se refleja en la Figura 42

```
  if ($("#myCheck1").is(":checked")){
  var latfinal= document.getElementById('latmin').value
  var latinicial= document.getElementById('latmax').value
  var longinicial= document.getElementById('lonmin').value
  var longfinal= document.getElementById('lonmax').value
  var val = document.getElementById('listapais').value;
  var val1= "rdf:type";
  var val2= "";
  var val3= "?objeto";
  var posit = "(?lat> "+latfinal+"  && ?lat < "+latinicial+") &&(?lon> "+longinicial+"  && ?lon < "+longfinal+")"
  var alt = document.getElementById('altura').value;

    if (val== 1) {
  val= "yago:ProvincialCapitalsInEcuador";
} else if (val== 2) {
  val= "yago:StatesOfTheUnitedStates";
  var val2= "?c dbpedia-owl:capital ?objeto. ";
  var val3= "?c";
}else if (val== 3) {
  val= "yago:GermanStateCapitals";
} else if (val== 4) {
   val= "dbpedia-owl:City"
}else if (val== 5) {
   val= "yago:CapitalsInEurope"
}else if (val== 6) {
   val= "category:Active_volcanoes"
   val1= "dcterms:subject"
}else if (val== 7) {
   val= "dbpedia-owl:Mountain"
}


var query= "SELECT ?name ?lon ?lat ?alt  WHERE {"+val3+" "+val1+" "+val+" . "+val2+" ?objeto rdfs:label ?name . ?
document.getElementById("cuadroconsulta").innerHTML = query;
sparqlQueryJson(query, endpoint, myCallbackalt, true)
}
```

Figura 42. Código JavaScript para crear la consulta en función de la elección del usuario (Pestaña altura)

Se puede observar que del mismo modo que ocurre en el menú anterior, sólo se ejecuta este pedazo de código si el *checkbox* del marco de trabajo que se quiere utilizar esta marcado a la hora de lanzar la consulta. Si está marcado, la aplicación almacena en 4 variables las latitudes y longitudes almacenadas en el marco, y crea otra una cadena de caracteres para definir en la consulta el ámbito de trabajo.

Seguidamente se almacena en otra variable el valor que se haya introducido para la altura mínima, y se crean otras, que servirán para modificar la consulta en función de la opción que se seleccione.

En función del elemento geográfico a representar que se haya seleccionado, entra en la sentencia condicional adecuada para asignar a las variables el valor que se necesita.

En la variable “query” se concatenan los valores de las variables. La forma de la consulta se así:

```
var query=
"SELECT ?name ?lon ?lat ?alt
WHERE {"+val3+" "+val1+" "+val+" .
 "+val2+" ?objeto rdfs:label ?name .
?objeto  geo:long   ?lon .
?objeto geo:lat    ?lat .
?objeto dbpedia-owl:elevation     ?alt.
FILTER(LANG(?name) = 'es' && (?alt > "+alt+") && "+posit+")}"
```

Una vez creada la cadena de caracteres adecuada, se envía a la citada función que crea la petición HTTP. Además de esto se creó una nueva función para procesar los JSON que se reciben con esta consulta, ya que son diferentes para cada consulta. Su código aparece en la Figura 43.

```
    function myCallbackalt(str) {
str = eval('(' + str + ')');
for(var i = 0; i<  str.results.bindings.length; i++) {
    var la = str.results.bindings[i].lat.value;
    var lo = str.results.bindings[i].lon.value;
    var name = str.results.bindings[i].name.value;
    var altura = str.results.bindings[i].alt.value;
    addMarker(lo,la,"Nombre: "+name+"</br>Altura: "+altura);
}
}
```

Figura 43. Código JavaScript para procesar el JSON (Pestaña altura)

En él se puede ver que además de latitud, longitud y nombre, se crea la variable altura, y se pasan los dos últimos como cadena de caracteres para que la función addMarker(), los coloqué en el *PopUp* de forma adecuada.

6.2.4. Consulta población

Al igual que la funcionalidad anterior, esta es capaz de mostrar de forma más atractiva al usuario la utilidad de datos enlazados de DBpedia. En este caso es capaz de filtrar por población (mayor, menor o entre dos valores), grupos de elementos geográficos como los municipios de Lugo, las provincias de España, las capitales de Sudamérica, las capitales de Oceanía, las capitales de Europa, las entidades de población etiquetadas como ciudad en DBpedia y las capitales de Asia.

La forma que tiene la interfaz de esta pestaña se muestra en la Figura 44.

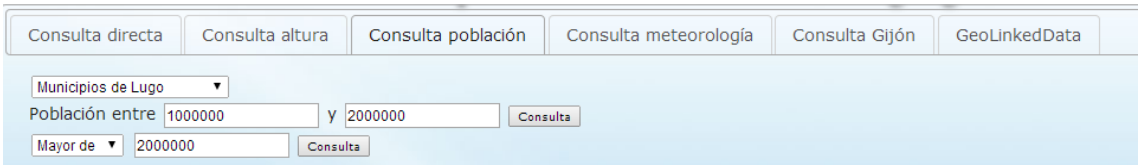


Figura 44. Interfaz de la pestaña "Consulta población"

En ella se debe seleccionar en la lista desplegable uno de los grupos de elementos geográficos que aparecen. Si se desean representar los elementos dentro de un determinado intervalo, se han de escribir en los cuadros de texto habilitados al efecto y pulsar el primer botón de consulta. En el caso de desear representar los elementos con una población mayor o menor que una determinada cifra, se debe seleccionar "mayor de" o "menor de", escribir en el cuadro de texto la cifra deseada y pulsar el segundo botón consultar.

Cada uno de los botones de consulta está asociado a una función diferente. La función del primero de ellos contiene el código que se muestra en la figura Figura 45:

```
function consultapobentre() {
    if (!$("#myCheck1").is(":checked") && !$("#myCheck2").is(":checked") && !$("#myCheck3").is(":checked")){
alert("Active marco de trabajo por favor");
}

if ($("#myCheck1").is(":checked")){
    var latfinal= document.getElementById('latmin').value
    var latinicial= document.getElementById('latmax').value
    var longinicial= document.getElementById('lonmin').value
    var longfinal= document.getElementById('lonmax').value
    var posit = "(?lat> "+latfinal+"  && ?lat < "+latinicial+") && (?lon> "+longinicial+"  && ?lon < "+longfinal+")"
    var mayor = document.getElementById('pobmay').value;
    var menor = document.getElementById('pobmen').value;
    var valcity = document.getElementById('listapob').value;
    if (valcity== 1) {
        valcity= "yago:MunicipalitiesInTheProvinceOfLugo";
    } else if (valcity== 2) {
        valcity= "yago:ProvincesOfSpain";}
      else if (valcity== 3) {
        valcity= "yago:CapitalsInSouthAmerica";
    } else if (valcity== 4) {
        valcity= "yago:CapitalsInOceania"
    }else if (valcity== 5) {
        valcity= "yago:CapitalsInEurope"
    }else if (valcity== 6) {
        valcity= "dbpedia-owl:City"
    }else if (valcity== 7) {
        valcity= "yago:CapitalsInAsia"
}
    var query= "SELECT ?name ?lon ?lat ?pob WHERE {?c a          "+valcity+" ; rdfs:label ?name ; geo:long   ?lon ; geo:lat
    document.getElementById("cuadroconsulta").innerHTML = query;
    sparqlQueryJson(query, endpoint, myCallbackpob, true)
}
```

Figura 45. Código JavaScript para crear la consulta en función de la elección del usuario en la pestaña "Consulta población" (Primer botón)

En esta parte de código se puede observar que primero se comprueba si todos los marcos de trabajo están desactivados. Si es así, envía un mensaje de error.

Si el primero de ellos está activado, coge los valores de latitud y longitud que tenga almacenados, los introduce en 4 variables y con ellas crea una nueva variable que concatena los valores de estas, formando la parte del FILTER de la consulta SPARQL que va a restringir su ámbito de actuación. Seguidamente, almacena en otras dos variables los valores introducidos para definir el intervalo de población con el que se quiere trabajar. Así mismo almacena en otra variable un valor que representa un determinado grupo de elementos geográficos (1 para los municipios de Lugo, 2 para las provincias de España…) según lo que se seleccionara en el desplegable correspondiente antes de pulsar el botón "consulta". Con el condicional que lo sigue se almacena la etiqueta correspondiente al elemente geográfico a consultar.

A continuación se concatenan todas las variables anteriores formando una cadena de caracteres con la forma de la consulta SPARQL a realizar:

```
var query =
SELECT ?name ?lon ?lat ?pob
WHERE {
?c a        "+valcity+" ;
rdfs:label ?name ;
geo:long   ?lon ;
 geo:lat    ?lat ;
dbpedia-owl:populationTotal     ?pob.
FILTER(  lang(?name)='es'  &&  (?pob>  "+menor+"  &&  ?pob<  "+mayor+")  &&
"+posit+")}
```

El valor de esta variable es mostrada en el cuadro azul y se llama a la función que envía la petición HTTP a DBpedia.

Como es lógico, este proceso se repetirá para los otros dos marcos, en caso de ser seleccionados, con la diferencia de que el valor tomado en las latitudes y longitudes será el correspondiente a cada marco de trabajo.

Al igual que en los casos anteriores, se debe crear una función que procese el JSON. En este caso debe ser como la que se muestra en la Figura 46.

```
function myCallbackpob(str) {
        str = eval('(' + str + ')');
        for(var i = 0; i<  str.results.bindings.length; i++) {
            var la = str.results.bindings[i].lat.value;
            var lo = str.results.bindings[i].lon.value;
            var name = str.results.bindings[i].name.value;
            var pob = str.results.bindings[i].pob.value;
            addMarker(lo,la,"Nombre: "+name+"</br>Poblaci&oacuten: "+pob);
        }
}
```

Figura 46. Código JavaScript para procesa el JSON (Pestaña altura)

Esta función hace exactamente lo mismo que su homóloga en la consulta de altura, pero almacenando el valor de población.

La función que tiene asociado el segundo botón “consultar” de esta pestaña aparece en la Figura 47.

```
//Población

function consultapob() {
    if (!$("#myCheck1").is(":checked") && !$("#myCheck2").is(":checked") && !$("#myCheck3").is(":checked")){
    alert("Active marco de trabajo por favor");
    }

if ($("#myCheck1").is(":checked")){
var latfinal= document.getElementById('latmin').value
var latinicial= document.getElementById('latmax').value
var longinicial= document.getElementById('lonmin').value
var longfinal= document.getElementById('lonmax').value
var posit = "(?lat> "+latfinal+"  && ?lat < "+latinicial+") && (?lon> "+longinicial+"  && ?lon < "+longfinal+")"

var val = document.getElementById('menormayorentre').value;
if (val ==1){
 val= ">";
 }else if (val ==2){
 val= "<";
 }
var valpob = document.getElementById('pobla').value;
var valcity = document.getElementById('listapob').value;
if (valcity== 1) {
    valcity= "yago:MunicipalitiesInTheProvinceOfLugo";
  } else if (valcity== 2) {
    valcity= "yago:ProvincesOfSpain";
  }else if (valcity== 3) {
    valcity= "yago:CapitalsInSouthAmerica";
  } else if (valcity== 4) {
     valcity= "yago:CapitalsInOceania"
  }else if (valcity== 5) {
     valcity= "yago:CapitalsInEurope"
  }else if (valcity== 6) {
     valcity= "dbpedia-owl:City"
  }else if (valcity== 7) {
     valcity= "yago:CapitalsInAsia"
  }
var query= "SELECT ?name ?lon ?lat ?pob WHERE {?c a   "+valcity+" ; rdfs:label ?name ; geo:long   ?lon ; geo:lat
document.getElementById("cuadroconsulta").innerHTML = query;
sparqlQueryJson(query, endpoint, myCallbackpob, true)

}
```

Figura 47. Código JavaScript para crear la consulta en función de la elección del usuario en la pestaña "Consulta población" (Segundo botón)

En esta parte de código, al igual que sucedía con la función anterior, se puede observar que primero se comprueba si todos los marcos de trabajo están desactivados. Si es así, envía un mensaje de error.

En este caso, si el primero de ellos está activado, coge los valores que tenga almacenados, los introduce en 4 variables y con ellas crea una nueva variable que concatena los valores de estas, formando la parte del "FILTER" de la consulta SPARQL. Seguidamente, con la ayuda de un condicional, se selecciona el símbolo de mayor o menor, según nuestra elección anterior en el desplegable correspondiente. Así mismo almacena en otra variable la cifra con la que se filtrará la población y en otra un valor que representa un determinado grupo de elementos geográficos (1 para los municipios de Lugo, 2 para las provincias de España…) según lo que se seleccionara en el desplegable asociado.

A continuación se muestra la forma de concatenar todas las variables anteriores de forma que sean una cadena de caracteres con la consulta SPARQL a realizar:

```
var query=
"SELECT ?name ?lon ?lat ?pob
WHERE {
?c a   "+valcity+" ;
rdfs:label ?name ;
geo:long   ?lon ;
geo:lat    ?lat ;
dbpedia-owl:populationTotal     ?pob.
FILTER(LANG(?name) = 'es' && (?pob"+val+" "+valpob+") && "+posit+" )}"
```

Esta variable es traspasada al cuadro azul para mostrar la consulta realizada y se llama a la función que envía la petición http a DBpedia.

Como sucedía en el caso anterior, toda lo que se encuentra dentro del condicional que actúa si el *checkbox* del marco de trabajo 1 está activado, debe ser triplicado para cada uno de los marcos restantes.

Al igual que en los casos anteriores, se debe crear una función que procese el JSON. Esta función es la misma que la citada anteriormente en la Figura 46.

6.2.5. Consulta meteorología

En este menú se tratarán de explotar los datos geográficos enlazados que se han sido introducidos en la iniciativa AemetLinkedData.

Al igual que sucedía en los dos menús anteriores, en este caso también se han introducido una serie de botones, desplegables y cuadros de texto para abstraer al usuario de todo lo que hay por detrás.

Este menú es capaz, primeramente de dibujar todas las estaciones meteorológicas disponibles de AEMET sobre el mapa y abrir un *PopUp* en cada chincheta dibujada que muestre el nombre de la estación y su código.

Además de esto, cuenta con la posibilidad de introducir el número de estación de la que se desee obtener algún dato, seleccionar con un desplegable la variable a consultar e introducir el número de valores registrados que se quieran visualizar de la citada variable.

El aspecto de este menú se refleja en la Figura 48.

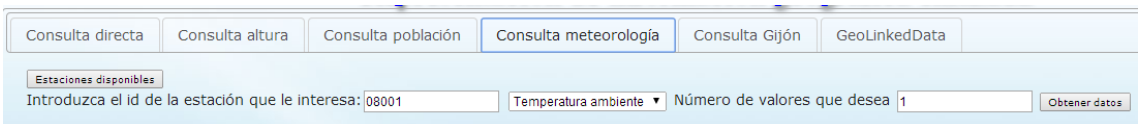


Figura 48. Interfaz de la pestaña "Consulta meteorología"

En él se pueden ejecutar dos consultas, la primera, la del botón "Estaciones disponibles", que traerá todas las estaciones del estado español y otra, la del botón "Obtener datos" que consultará algún valor de la variable que se seleccione y de la estación que se introduzca en el primer cuadro de texto.

El código de la función que se ejecuta cuando se pulsa el primer botón aparece en la Figura 49.

```
function consultaestaciones(){

var query = "PREFIX aemetonto:   <http://aemet.linkeddata.es/ontology/> PREFIX geo:  <http://www.w3.org/2003/01/geo/wgs84_pos%23>
document.getElementById("cuadroconsulta").innerHTML = query;
sparqlQueryJsonmeteo(query)

}
```

Figura 49. Código JavaScript que almacena la primera consulta de la pestaña "Consulta Meteorología"

En él se especifica la consulta a realizar, que en este caso es fija, se hace que aparezca en el cuadro azul de la interfaz y se envía la petición a otra función que enviará la petición HTTP. Esta función aparece reflejada en la Figura 50.

```
function sparqlQueryJsonmeteo(queryStr) {
    $.get("http://localhost:8080/openrdf-sesame/repositories/AEMET?query="+queryStr+"&Accept=application%2Fsparql-results%2Bjson&callback=?&format=application",
    {}, function(data) {
   myCallbackest(data);
    console.log('data = ', data);

 });
}
```

Figura 50. Código JavaScript que envía la petición HTTP GET a Sesame

En este caso se concatena la consulta con la raíz del *endpoint* correspondiente a nuestro servidor y al repositorio seleccionado, además de al formato de salida de los datos (JSON). Toda esa cadena de caracteres es enviada en una petición GET. Si la respuesta es positiva, se almacena el resultado en la variable "data" y se ejecuta la función que procesará el JSON que se ha obtenido.

El código de esta función que procesará el JSON tiene el código que se muestra en la Figura 51.

```
function myCallbackest(str) {
if (!$("#myCheck1").is(":checked") && !$("#myCheck2").is(":checked") && !$("#myCheck3").is(":checked")){
      for(var i = 0; i<  str.results.bindings.length; i++) {
          var la = str.results.bindings[i].lat.value;
          var lo = str.results.bindings[i].lon.value;
          var name = str.results.bindings[i].nombre.value;
          var code = str.results.bindings[i].codigo.value

          addMarker(lo,la,"Nombre estaci&oacuten:"+name  +"   C&oacutedigo= "+code);
       }
   }

       if ($("#myCheck1").is(":checked")){
      var latfinal= document.getElementById('latmin').value
      var latinicial= document.getElementById('latmax').value
      var longinicial= document.getElementById('lonmin').value
      var longfinal= document.getElementById('lonmax').value
      for(var i = 0; i<  str.results.bindings.length; i++) {
          var la = str.results.bindings[i].lat.value;
          var lo = str.results.bindings[i].lon.value;
          var name = str.results.bindings[i].nombre.value;
          var code = str.results.bindings[i].codigo.value
          if (longinicial>0 && longfinal>0){
          if (( la>latfinal && la<latinicial ) &&  (lo<longfinal && lo>longinicial)){
          addMarker(lo,la,"Nombre estaci&oacuten:"+name  +"   C&oacutedigo= "+code);}
              } else if (longinicial<0 && longfinal<0){
              if (( la>latfinal && la<latinicial ) &&  (lo>longfinal && lo<longinicial)){
          addMarker(lo,la,"Nombre estaci&oacuten:"+name  +"   C&oacutedigo= "+code);}

              } else if (longinicial<0 && longfinal>0){
              if (( la>latfinal && la<latinicial ) && (lo>longfinal && lo>longinicial)){
          addMarker(lo,la,"Nombre estaci&oacuten:"+name  +"   C&oacutedigo= "+code);}
              }
       }
       }
```

Figura 51. Código JavaScript que procesa el JSON y aplica los marcos en la pestaña "Consulta meteorología"

Como se puede observar, si permanecen sin marcar todos los marcos de trabajo, la función procesa todas los datos obtenidos del JSON, sin embargo, de pulsarse alguno de los *checkbox* que activan los marcos de trabajo, se filtran los resultados en función de su latitud y longitud con la ayuda de una serie de condicionales anidados, escogiendo solo aquellas que estén dibujadas en el marco. Esto se aplicará para cada uno de los marcos.

Al finalizar cada condicional, se ejecuta la función addMarker(), que dibujara en el mapa las chinchetas correspondientes.

Una vez que se hayan dibujado las chinchetas de las estaciones de interés, haciendo clic en cada una de ellas se puede consultar su nombre y código de estación. En la Figura 52 se muestra un ejemplo de esto.

Figura 52. PopUp de un marcador mostrando el código de estación

De este modo ya se puede conocer el código de la estación que podría interesar al usuario. Sabiendo esto, se pueden consultar los datos de las últimas variables registradas e incluidas en el proyecto AemetLinkedData para esa estación. Esto lo consigue con el pedazo de código de la Figura 53.

```
function datosestacion(){

var codestac = document.getElementById('codesta').value;
var valnumdat = document.getElementById('numdatos').value;
var prop = document.getElementById('parametro').value;

if (prop==1)
{prop= "<http://aemet.linkeddata.es/resource/TemperatureAmbientProperty/TA>"}
else if (prop==2)
{prop= "<http://aemet.linkeddata.es/resource/PrecipitationAmbientProperty/PREC>"}
else if (prop==3)
{ prop= "<http://aemet.linkeddata.es/resource/PressureAmbientProperty/PRES>"
}

var query = "PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema%23> PREFIX xsd:  <http://www.w3.org/2001/XMLSchema%23>
document.getElementById("cuadroconsulta").innerHTML = query;
sparqlQueryJsonprop(query)

function sparqlQueryJsonprop(queryStr) {
    $.get("http://localhost:8080/openrdf-sesame/repositories/AEMET?query="+queryStr+"&Accept=application%2Fsparql-resu
    myCallbackprop(data);
    console.log('data = ', data);

  });
}
```

Figura 53. Código JavaScript que crea la consulta a una estación concreta

En él primeramente se registran los valores escritos en los cuadros de texto y la opción señalada en el desplegable en el momento de ejecutar la consulta. Seguidamente se asigna el valor que debe tomar la variable a consultar para formar la consulta SPARQL definitiva. Finalmente se forma una cadena de caracteres con estas variables que tiene la siguiente forma:

PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema%23>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema%23>
PREFIX ssn: <http://purl.oclc.org/NET/ssnx/ssn%23> PREFIX aemet: <http://aemet.linkeddata.es/ontology/>
PREFIX w3ctime:<http://www.w3.org/2006/time%23>
PREFIX geo: <http://www.w3.org/2003/01/geo/wgs84_pos%23>
SELECT ?nombre ?lat ?lon ?nameprop ?valueprop
WHERE {
?estacion aemet:indsinop '"+codestac+"'.
?estacion aemet:stationName ?nombre.
?estacion geo:location ?loc.
?loc geo:lat ?lat.
?loc geo:long ?lon.
?obs ssn:observedBy ?station .
?obs ssn:observedProperty ?prop.
?prop rdfs:label ?nameprop.
?obs aemet:valueOfObservedData ?valueprop .
FILTER(?prop = "+prop+") }
LIMIT "+valnumdat+"

Esta consulta almacenada en una variable, es traspasada al cuadro azul y enviada a una función que realizará la petición HTTP. Así mismo, esta nueva función envía los datos recibidos de la petición a otra función que procesará el JSON. En este caso esta función tiene la forma que aparece en la Figura 54.

```
function myCallbackprop(str) {
    console.log('lat'+str)
    var valor = [];

        for(var i = 0; i<  str.results.bindings.length; i++) {
            var la = str.results.bindings[i].lat.value;
            var lo = str.results.bindings[i].lon.value;
            var name = str.results.bindings[i].nombre.value;
            var prop = str.results.bindings[i].nameprop.value;

            valor[i] = str.results.bindings[i].valueprop.value;
            p=i+1;

            }
            addMarker(lo,la,"Nombre estaci&oacuten:"+name  +"   Propiedad= "+prop+"</br></br><tr>"+valor+"<tr>");
    }

}
```

Figura 54. Código JavaScript que procesa el archivo JSON en la pestaña "Consulta meteorología"

En ella se procesa el JSON con un bucle y se envían los datos a otra función que dibujará la chincheta en el mapa, mostrando en su PopUp los datos de la variable seleccionada.

6.2.6. Consulta datos Gijón

En el caso de esta pestaña, se van a explotar datos enlazados del Ayuntamiento de Gijón, almacenados en un repositorio propio, como se ha explicado en el apartado 5.4.

Debido a que la información enlazada disponible de los servicios de Gijón no está demasiado enriquecida (Figura 23), las operaciones que se pueden realizar con ellas son bastante reducidas.

Los servicios del ayuntamiento que se van a permitir consultar la aplicación se corresponden con aparcamientos, zonas de ocio nocturno, bancos, bibliotecas, boleras, camping, campos de fútbol, cines y comercios.

La interfaz de esta funcionalidad tiene el aspecto que se muestra en la Figura 55.

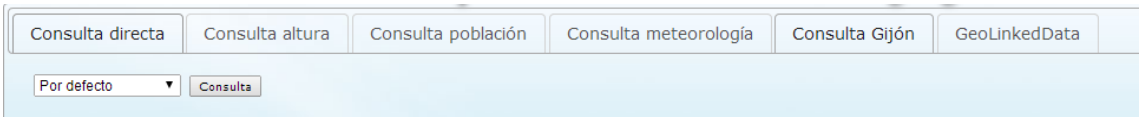


Figura 55. Interfaz de la pestaña "Consulta Gijón"

En el desplegable se puede seleccionar alguno de los servicios mencionados. Si se pulsa el botón "Consulta" se ejecuta una función que tiene el código que aparece en la Figura 56.

```
function datosgijon(){

var servicio = document.getElementById('datogijon').value;
if (servicio=="Aparcamientos")
{
servicio= "aparcamientos";
}
else if (servicio=="Pubs")
{servicio= "pubs"}
else if (servicio=="Bancos")
{ servicio= "bancoscajeros"}
else if (servicio=="Bibliotecas")
{servicio= "bibliotecas"}
else if (servicio=="Boleras")
{ servicio= "boleras"}
else if (servicio=="Camping")
{servicio= "camping"}
else if (servicio=="Futbol")
{ servicio= "camposfutbol"}
else if (servicio=="Cines")
{servicio= "cines"}
else if (servicio=="Comercios")
{ servicio= "comercios"}

var query = "PREFIX geo: <http://www.geonames.org/ontology%23>  PREFIX cal: <http://www.w3.org/2002/12/cal%23>
document.getElementById("cuadroconsulta").innerHTML = query;
sparqlQueryJsongijon(query)
```

Figura 56. Código JavaScript que crea la consulta en función de la elección del usuario en la pestaña "Consulta Gijón"

En el código de la Figura 56, se almacena en una variable el servicio seleccionado y con la ayuda de un condicional se asigna a la misma variable, el valor válido para hacer la consulta SPARQL. La forma de la consulta será el siguiente:

```
PREFIX geo: <http://www.geonames.org/ontology%23>
PREFIX cal: <http://www.w3.org/2002/12/cal%23>
PREFIX dc: <http://purl.org/dc/elements/1.1%23>
SELECT ?nombre ?localizacion
FROM <http://localhost:8080/openrdf-sesame/DatosGijon/"+servicio+".rdf>
WHERE{
?datasetURI geo:name ?nombre .
?datasetURI cal:location ?localizacion. }
```

Como se puede ver, la consulta es prácticamente igual para todos los servicios, excepto por la URI del contexto del que se desean extraer los datos.

Una vez creada la consulta en una variable, se procede de igual modo que en los casos anteriores. Se envía por petición GET y se recibe un JSON, y este JSON es procesado por otra función, que a su vez envía cada uno de los datos extraídos a otra función que dibuja en el mapa los objetos procesados.

En el caso de esta pestaña, se han seleccionado chinchetas diferentes según el servicio a representar. Para lograr esto se han introducido los siguientes condicionales (Figura 57) en la función que añade los *markers* (Figura 40).

```
        if (servicio=="Aparcamientos")
}{
var icon = new OpenLayers.Icon("http://google-maps-icons.googlecode.com/files/parking.png",iconSize, iconOffset);}
else if (servicio=="Pubs")
{var icon = new OpenLayers.Icon("http://google-maps-icons.googlecode.com/files/cocktail.png",iconSize, iconOffset);}
else if (servicio=="Bancos")
{ var icon = new OpenLayers.Icon("http://google-maps-icons.googlecode.com/files/bankeuro.png",iconSize, iconOffset);}
else if (servicio=="Bibliotecas")
{var icon = new OpenLayers.Icon("http://google-maps-icons.googlecode.com/files/library-uni.png",iconSize, iconOffset);}
else if (servicio=="Boleras")
{var icon = new OpenLayers.Icon("http://google-maps-icons.googlecode.com/files/bowling.png",iconSize, iconOffset);}
else if (servicio=="Camping")
{var icon = new OpenLayers.Icon("http://google-maps-icons.googlecode.com/files/tent.png",iconSize, iconOffset);}
else if (servicio=="Futbol")
{var icon = new OpenLayers.Icon("http://google-maps-icons.googlecode.com/files/soccer2.png",iconSize, iconOffset);}
else if (servicio=="Cines")
{var icon = new OpenLayers.Icon("http://google-maps-icons.googlecode.com/files/cinema.png",iconSize, iconOffset);}
else if (servicio=="Comercios")
{var icon = new OpenLayers.Icon("http://google-maps-icons.googlecode.com/files/supermarket.png",iconSize, iconOffset);}
else if (servicio==10)
{var icon = new OpenLayers.Icon("http://www.openstreetmap.org/openlayers/img/marker.png",iconSize, iconOffset);}
```

Figura 57. Código JavaScript que sitúa un icono diferente en función de la elección del usuario.

6.2.7. Consulta Geo-LinkedData

Al igual que sucedía con los datos de Gijón, los datos que el Ontology Engineering Group transformó a datos enlazados de algunos de los proyectos del IGN, cuentan con muy poca información, con lo que su funcionalidad se restringirá a simplemente representar algunos de los elementos sin ningún tipo de filtro, excepto el marco de trabajo.

De todo el conjunto de datos que existe en esta iniciativa, se seleccionaron elementos de hidrografía. El criterio de esta selección ha sido simplemente por el motivo de trabajar con un grupo homogéneo de información. Por lo tanto, los elementos que se podrán representar son los nacimientos de los ríos, las piscinas, las playas, las lagunas, los pozos, los pantanos, los mares, las marismas y los faros.

El aspecto de la interfaz de esta parte de la aplicación se muestra en la Figura 58.

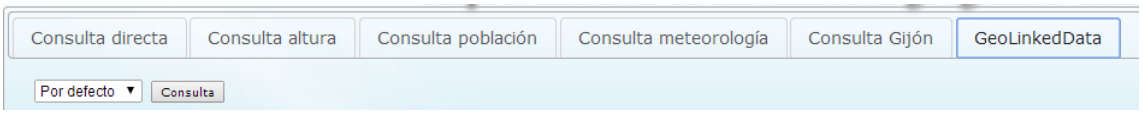


Figura 58. Interfaz de la pestaña "GeoLinkedData"

En ella se pide seleccionar alguno de los mencionados elementos hidrográficos, para realizar la consulta en función de la elección.

Al pulsar el botón "Consultar" se ejecuta la función que aparece en la Figura 59.

```
function datosign(){

var concepto = document.getElementById('datoign').value;

if (concepto==1)
{concepto= "Nacimiento";}
else if (concepto==2)
{concepto= "Piscina"}
else if (concepto==3)
{ concepto= "Playa"}
else if (concepto==4)
{concepto= "Laguna"}
else if (concepto==5)
{ concepto= "Pozo"}
else if (concepto==6)
{concepto= "Embalse"}
else if (concepto==7)
{ concepto= "Mar"}
else if (concepto==8)
{concepto= "Marisma"}
else if (concepto==9)
{ concepto= "Faro"}

var endpoint = "http://localhost:8080/openrdf-sesame/repositories/IGN";
var query = "PREFIX geo: <http://www.w3.org/2003/01/geo/wgs84_pos%23> PREFIX geoes: <http://geo.linkeddata.es/ontology/> SELECT
document.getElementById("cuadroconsulta").innerHTML = query;
sparqlQueryJsongeolinked(query, endpoint, myCallbackgeolinked, true)
}

function sparqlQueryJsongeolinked(queryStr) {
    $.get("http://localhost:8080/openrdf-sesame/repositories/IGN?query="+queryStr+"&Accept=application%2Fsparql-results%2Bjson&c
    myCallbackgeolinked(data);
  });
}
```

Figura 59. Código JavaScript que crea la consulta en función de la elección del usuario en la pestaña "GeoLinkedData"

En ella, al igual que en las anteriores, se almacena el valor asociado a la selección que se haya hecho en el desplegable y se asocia con un condicional el valor que debe tener en la consulta SPARQL.

Seguidamente, se concatena la variable con el resto de la consulta formando una consulta SPARQL como la siguiente:

PREFIX geo: <http://www.w3.org/2003/01/geo/wgs84_pos%23>
PREFIX geoes: <http://geo.linkeddata.es/ontology/>
SELECT ?nombre ?lat ?lon {
?concepto rdf:type geoes:"+concepto+".
?concepto rdfs:label ?nombre.
?concepto geo:geometry ?loc.
?loc geo:lat ?lat.
?loc geo:long ?lon. }

Esta consulta es representada en el cuadro azul y se envía a otra función que hará la petición HTTP al repositorio correspondiente. De esta petición se obtendrá el archivo JSON que será procesado por otra función, que es la que aparece en la Figura 60.

```
function myCallbackgeolinked(str) {
if (!$("#myCheck1").is(":checked") && !$("#myCheck2").is(":checked") && !$("#myCheck3").is(":checked")){
        for(var i = 0; i<  str.results.bindings.length; i++) {
            var la = str.results.bindings[i].lat.value;
            var lo = str.results.bindings[i].lon.value;
            var name = str.results.bindings[i].nombre.value;
            addMarkergeolinked(lo,la,"Nombre: " +name)
        }
    }

    if ($("#myCheck1").is(":checked") ){
       var latfinal= document.getElementById('latmin').value
       var latinicial= document.getElementById('latmax').value
       var longinicial= document.getElementById('lonmin').value
       var longfinal= document.getElementById('lonmax').value
        for(var i = 0; i<  str.results.bindings.length; i++) {
            var la = str.results.bindings[i].lat.value;
            var lo = str.results.bindings[i].lon.value;
            var name = str.results.bindings[i].nombre.value;
            if (longinicial>0 && longfinal>0){
                if (( la>latfinal && la<latinicial ) &&  (lo<longfinal && lo>longinicial)){
            addMarkergeolinked(lo,la,"Nombre: " +name)}
            } else if (longinicial<0 && longfinal<0){
                if (( la>latfinal && la<latinicial ) &&  (lo>longfinal && lo<longinicial)){
            addMarkergeolinked(lo,la,"Nombre: " +name)}
                }
        }
    }
```

Figura 60. Codigo JavaScript que procesa el JSON obtenido en la pestaña “GeoLinkedData”

En el pedazo de código de la Figura 60, se puede observar como a la hora de procesar el JSON se comprueba primero si los marcos de trabajo están activados. De no ser así, se procesa toda la información que se obtiene, es decir, se envía a la función que dibuja los marcadores todos los datos que reciba del JSON. Si alguno de los marcos estuviera activado, gracias al conjunto de condicionales anidados dispuestos en la función, restringiría la colocación de marcadores en función de las coordenadas almacenadas en el respectivo marco.

7. Resultados

Tras todo el proceso de desarrollo tanto interno como externo, se ha obtenido un *software* capaz de visualizar datos geográficos enlazados provenientes de diferentes fuentes de información.

Se ha logrado integrar en una misma aplicación datos geográficos enlazados provenientes de DBpedia, del Ayuntamiento de Gijón, de AEMET y del IGN.

Para conseguir los datos enlazados se ha trabajado de 3 maneras posibles, consultando los servidores de DBpedia mediante petición HTTP directa, almacenando y consultando un repositorio propio con ayuda de SESAME y consultando servidores externos a través del citado *software*.

Así mismo se ha logrado implementar una interfaz amigable que abstrajera todo el proceso de manipulación de consultas SPARQL y de datos enlazados.

8. Conclusiones y trabajos futuros

El objetivo principal de este trabajo ha sido mostrar las posibilidades de utilización de los datos enlazados abiertos a la hora de su aplicación en el ámbito geográfico. Se ha realizado una aplicación web a modo de prueba de concepto, capaz de visualizar información geográfica proveniente de datos enlazados de diversas fuentes.

Se ha conseguido trabajar a partir de un repositorio privado de datos y a partir de servidores externos o *endpoints* SPARQL y se han explotado los datos enlazados con ayuda del lenguaje SPARQL pudiendo mostrar el potencial de los datos enlazados en algunas partes de la aplicación.

Además de realizar la explotación de servidores externos de datos enlazados con peticiones directas HTTP, se ha utilizado software especializado para ello. Tras analizar diversas plataformas de tratamiento, explotación y consumo de datos enlazados, se decidió utilizar Sesame por ser de código abierto, fácil instalación, más veloz que sus competidores y con una interfaz simple y cómoda de utilizar.

El desarrollo de la aplicación se ha llevado a cabo en lenguaje JavaScript, empleando las librerías de Openlayers y Jquery, así como la técnica de desarrollo web AJAX, permitiendo una utilización interactiva de la información geográfica enlazada. De esta forma se ha llevado a cabo la importante labor de abstraer al usuario de todo el proceso interno de tratamiento de los datos enlazados

Algunos de los contrapuntos que se han encontrado en el desarrollo del proyecto han sido:

- La fuente de datos enlazados más completa que se ha encontrado ha sido DBpedia, permitiendo realizar consultas más complejas y por consiguiente más enriquecidas. Con las otras iniciativas que se han tenido en cuenta, las operaciones han sido más simples, ya que los datos disponibles contenían una información muy limitada.
- La fiabilidad y exactitud de la visualización final está siempre supeditada a la calidad de los datos, la cual es un poco deficiente en algunos conceptos de DBpedia e incompleta en el resto de iniciativas que se tuvieron en cuenta en el estudio.
- El comportamiento de la aplicación depende del funcionamiento de los servidores externos. Un ejemplo de esto es que alguno de los *endpoint* (DBpedia) consultados sufre un comportamiento errático en algunos momentos, dejando de estar operativo en intervalos de tiempo aislados, lo que hace que no se puedan lanzar las consultas SPARQL y afecte al comportamiento final de la aplicación.

A la hora de trabajar con datos enlazados se ha comprobado es muy importante conocer cuál es su forma, que información contiene, cuales son las etiquetas que utiliza y como de completos son.

Actualmente existe una tecnología desarrollada y completamente operativa que permite enlazar información y de ese modo enriquecerla y hacerla más útil, pero su utilización por el momento es muy limitada. La evolución relacionada con la implantación general de los

datos enlazados, con las ventajas que estos suponen, será con toda probabilidad, un campo de acción en futuras investigaciones.

Uno de los principales problemas que se deberían solucionar respecto a los datos enlazados es lo relativo a su calidad. Para la mejora en la utilización de datos abiertos y la consecución de su generalización global es importante la realización de estudios y controles de calidad de estos datos, ya que en la actualidad la ausencia (omisión o comisión) de cierta información en los datos enlazados es un condicionante muy importante.

Un punto de continuación con el presente estudio sería realizar la explotación de datos enlazados en más servidores y aumentar el número de repositorios de datos RDF propios.

Otra futura vía de investigación podría ser el intento de implementación de un módulo que permita introducir nuevos datos y borrar los existentes de una forma intuitiva para el usuario, a un repositorio de datos enlazados almacenados en un servidor propio.

El camino que deberían seguir las diferentes iniciativas con *endpoints* SPARQL disponibles es a federar sus *endpoints* con las de otras iniciativas y así poder conseguir datos de diversas iniciativas en una sola consulta.

Adaptar los *endpoints* al lenguaje de consulta GeoSPARQL, ya que la actualidad no existen muchos que permitan su uso. Explotar una base de datos enlazados propia con este lenguaje de consulta.

9. Referencias

[1]. SINTEF. "Big Data, for better or worse: 90% of world's data generated over last two years." ScienceDaily. ScienceDaily, 22 May 2013. <www.sciencedaily.com/releases/2013/05/130522085217.htm>.

[2]. Oliveros, J.H.; Valencia, J.M.; Torres, J.P.(2011): " Formato de intercambio de información geográfica entre dispositivos embebidos". *Sistemas, Cibernética e informática*. Vol.: 8 Número: 2 ISSN 1960-8627 pp.: 38-42.

[3]. W3C: http://www.w3c.es/Divulgacion/GuiasBreves/WebSemantica

[4]. Datos de la web semántica: http://www.maestrosdelweb.com/editorial/web-semantica-y-sus-principales-caracteristicas/

[5]. SmartOpenData Grant no.: 603824: "Requirements of the SmarOpenData Infrastructure" Deliverable D2.1 :: Public.

[6]. SmartOpenData Grant no.: 603824: "Dissemination Plan" Deliverable D7.6 :: Public.

[7]. DBpedia: http://dbpedia.org/About

[8]. Geonames Ontology: http://www.geonames.org/ontology/documentation.html

[9]. Ordnance Survey: http://www.ordnancesurvey.co.uk/education-research/research/linked-data-web.html

[10]. GeoLinkedData.es : http://red.linkeddata.es/web/guest/about-us

[11]. Iniciativa AEMETLinkedData: http://aemet.linkeddata.es/sparql.html

[12]. Datos enlazados de Gijón: https://datos.gijon.es/page/12217-servicio-sparql

[13]. 4Store: http://4store.org/

[14]. Openlink Virtuoso: http://virtuoso.openlinksw.com/

[15]. Openrdf Sesame: http://www.openrdf.org/

[16]. Oracle Spatial 11g: http://www.oracle.com/technetwork/es/documentation/317501-esa.pdf

[17]. Allegrograph: http://franz.com/agraph/allegrograph/

[18]. Jena2: https://jena.apache.org/

[19]. Openlayers: http://openlayers.org/

[20]. JQuery: http://jquery.com/

[21]. Tutoriales de AJAX, JQuery y JavaScript: http://www.w3schools.com/

[22]. Ontology Engineering Group: http://www.oeg-upm.net/

[23]. Ayuda JavaScript: http://www.wextensible.com/como-se-hace/pestanyas-tabs/con-javascript.html

[24]. Endpoint de DBpedia: http://dbpedia.org/sparql

[25].Tutorial de Sesame, Mariano Rodriguez Muro, Universidad de Bozen-Bolzano: http://pt.slideshare.net/marianomx/4-sesame

[26].Tutorial Sesame: http://greco.ppgi.ufrj.br/lodbr/index.php/principal/tutorial/sesame/

[27].Instalación Sesame: http://dookieweb.blogspot.com.es/2010/11/openrdf-sesame-i-instalacion.html

[28].Vilches-Blázquez, L.M.; Corcho, O.; González, A.; Villazon-Terrazas, B.; Gómez, J.M. (2014): "Combinando Open y Linked Data en el marco de las Smart Cities: Un caso de uso sobre alquiler de bicicletas". *UPM- Ontology Engineering Grour (OEG) e iSOCO*. Id Exp: IPT-20111006 Páginas: 7.

[29].Castelló, A. (2006): "Web semántica: RDF y SGBD que lo sopotan" Trabajo Final de carrera I.T. Informática Gestión. Páginas: 60

[30].Vilches-Blázquez, L.M; Sevilla, C.; Villalón, M.; Rodríguez, F.; Gómez-Perez, A. (2013): "Combinando Linked Data con servicios geoespaciales". *IV Jornadas Ibéricas de Infraestructuras de Datos Espaciales*. Páginas: 10.

[31].Silva, M.; Ceregatti, J.S.; Cintra, D.; Bauzer, C. (2011):"Using linked data to extract geo-knowledge". Proceedings XII GEOINFO, Campos do Jordao, Brazil. pp.:111-116.

[32].Sarango, D. (2011): "Publicación de datos universitarios observando los principios de Linked Data". Universidad Técnica Particular de Loja. Trabajo fin de carrera I. Sistemas Informáticos y Computación. Páginas: 198.

[33].De la Torre, P. (2009):"Almacenes de Datos para la Web Semántica". Universidad de Sevilla. Poyecto Fin de Carrera. Ingeniería Informática. Páginas: 101.

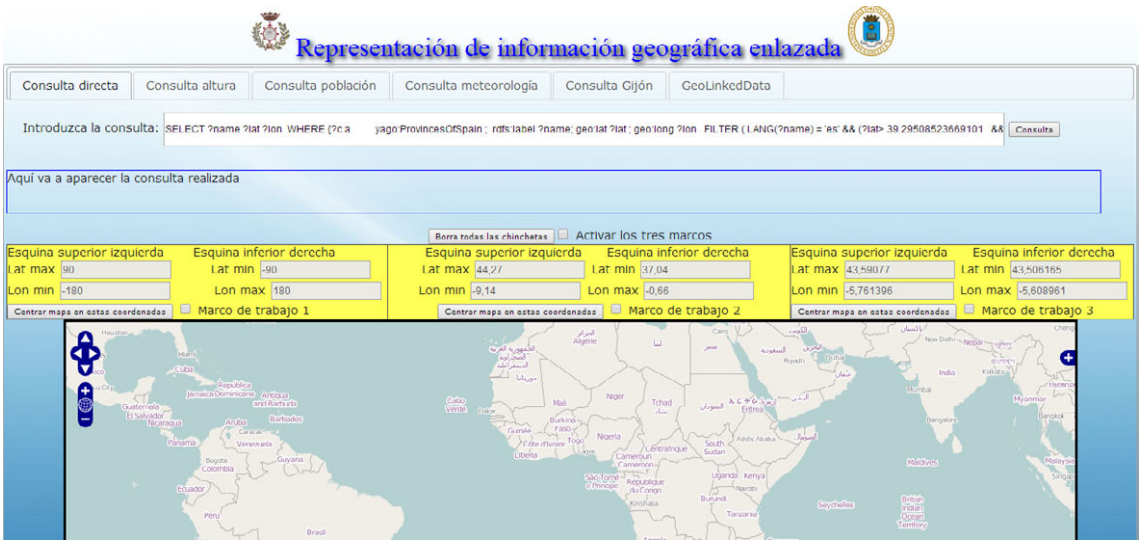
[34].Índice de endpoints SPARQL: http://www.w3.org/wiki/SparqlEndpoints

Anexos

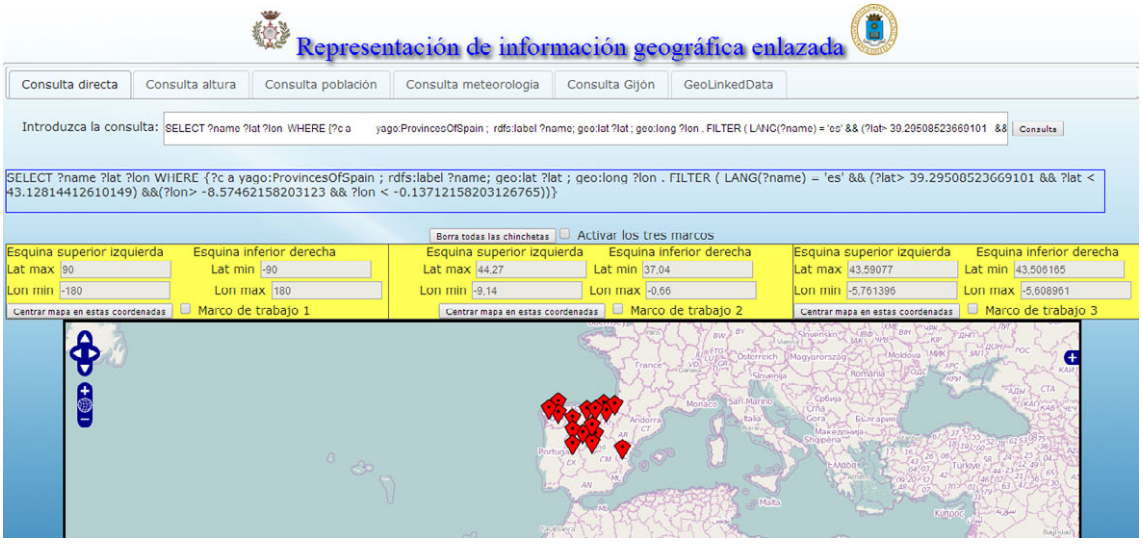
Anexo 1: Manual del usuario

En este anexo se mostrará un manual del usuario que permitirá a cualquier persona familiarizarse fácilmente con la utilización de la aplicación "Representación de la información geográfica enlazada".

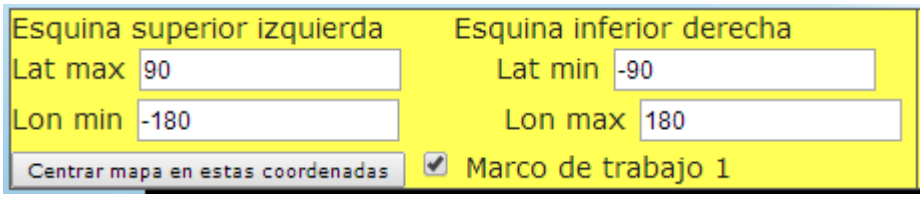
Esta es la interfaz que aparece al iniciar el programa:



Por defecto aparece seleccionada la pestaña "Consulta directa", en ella se permite realizar cualquier consulta SPARQL a DBpedia que devuelva como resultado el nombre, latitud y longitud de los elementos que se quieran visualizar. Por defecto aparece una consulta que devuelve las provincias de España en una de terminada latitud y longitud. Si se lanza la consulta devuelve lo siguiente:

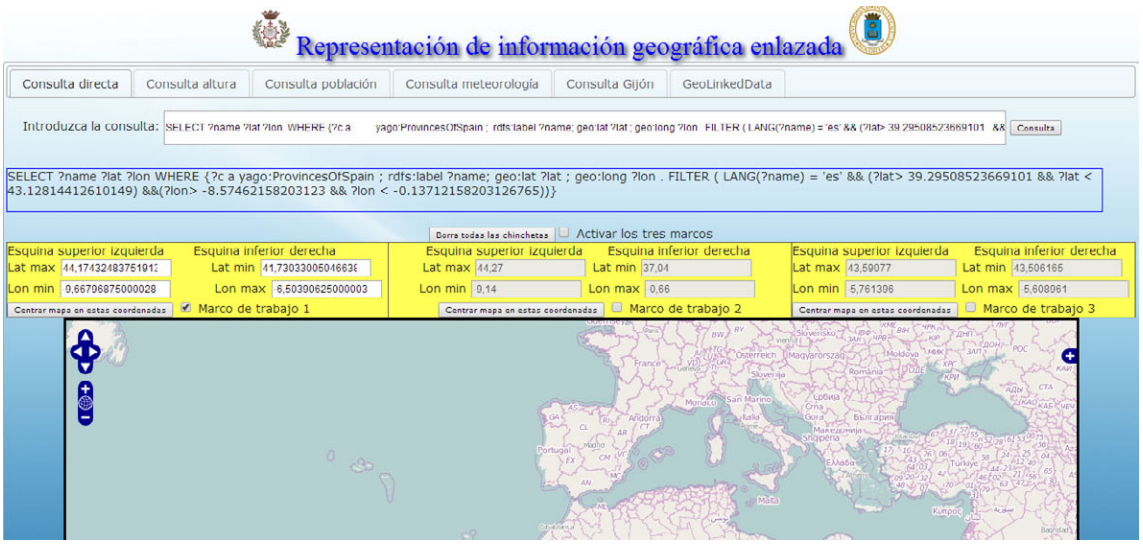


Si se desea filtrar el marco de trabajo para realizar la consulta, se debe activar alguno de ellos:

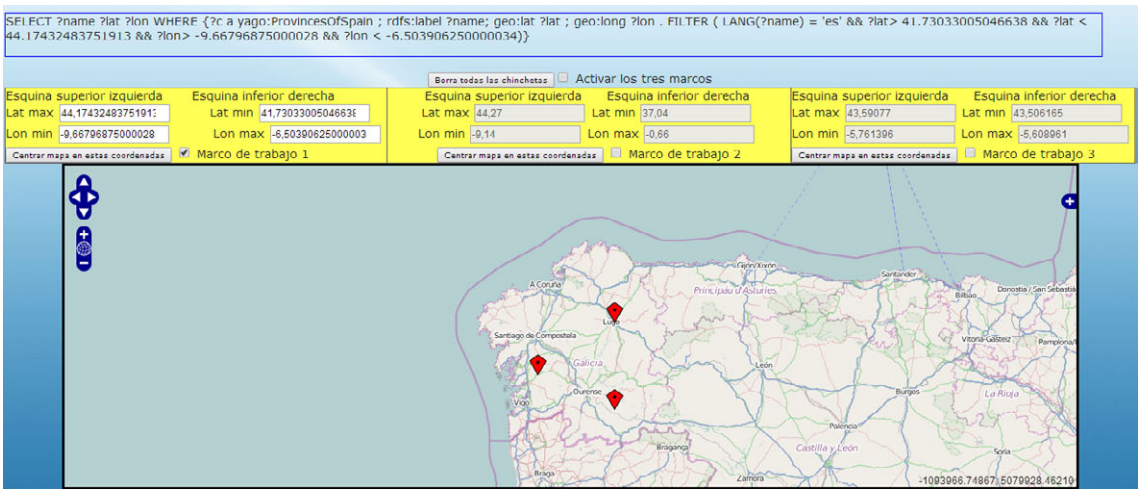


Al hacer un primer clic con un marco activado, aparecerán las coordenadas de la esquina superior izquierda del recuadro a definir como ámbito de trabajo y haciendo un segundo clic se puede definir la esquina inferior derecha. De esta forma se fijaran las coordenadas del marco.

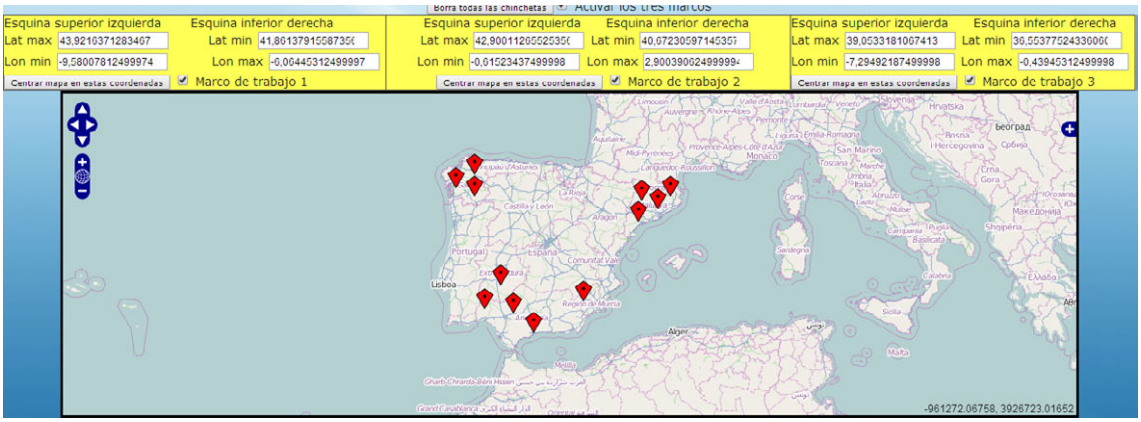
Para que la aplicación del marco de trabajo se haga efectiva, se debe mantener activo alguno de ellos. En este caso se han seleccionado unas coordenadas que encuadran la Comunidad autónoma de Galicia:



Al pulsar el botón “Consulta” se efectúa la misma consulta pero teniendo en cuenta el marco de trabajo seleccionado. Además de eso si se pulsa el botón “Centrar mapa en estas coordenadas”, el mapa se coloca en el recuadro definido anteriormente:



Además de esto también se pueden tener en cuenta tres marcos a la vez, situando las coordenadas deseadas en ellos. En la siguiente imagen se muestra la consulta focalizada en Galicia, Cataluña y Andalucía y sus resultados:



Se debe tener la precaución de no mantener otros marcos activados en el momento de capturar las coordenadas, si no se capturarán en varios marcos simultáneamente.

Como ejemplo, algunas consultas que se podría introducir para probar el potencial de los datos enlazados de DBpedia podrían ser:

- Situación de todos los campos de futbol de equipos de primera división:

```
SELECT ?name ?lat ?lon
WHERE {
?subject dbpedia-owl:league          dbpedia:La_Liga.
?subject rdfs:label ?name.
?subject dbpedia-owl:ground ?campo.
?campo geo:lat ?lat.
?campo geo:long ?lon.

FILTER ( LANG(?name) = 'es')}
```

- Lugar de nacimiento de los premios noveles de la Paz:

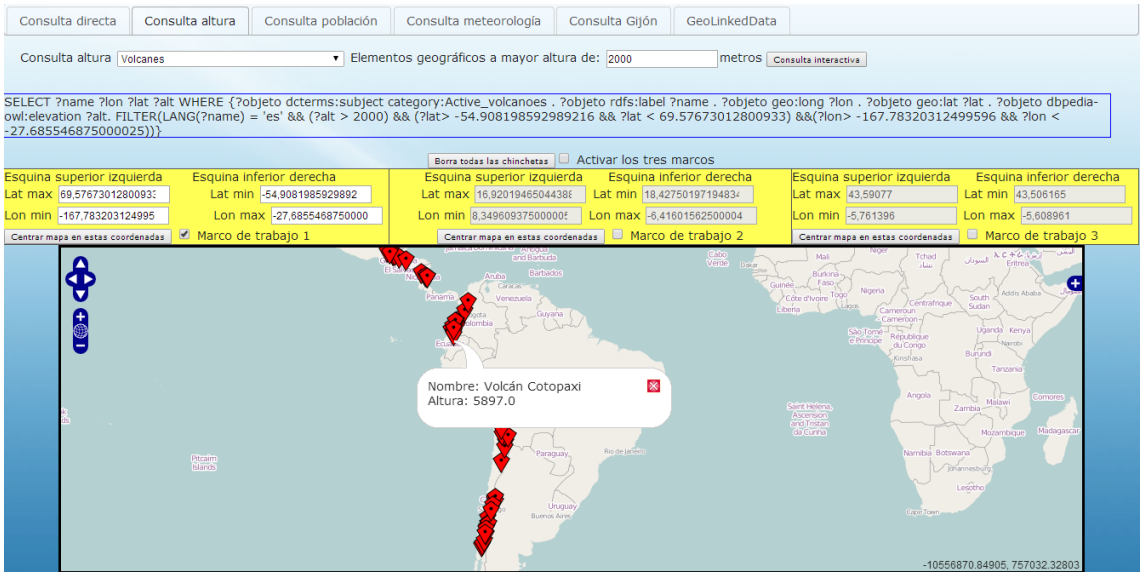
```
SELECT ?name ?lat ?lon
WHERE {
?subject rdf:type yago:NobelPeacePrizeLaureates.
?subject rdfs:label ?name.
?subject dbpprop:birthPlace ?lugar.
?lugar geo:lat ?lat.
?lugar geo:long ?lon.

FILTER ( LANG(?name) = 'en')}.
```

Existe además disponible un botón para borrar todas las chinchetas que estén sobre el mapa (Aunque al realizar una nueva consulta las antiguas se borran automáticamente) y un checkbox que activa o desactiva automáticamente todos los marcos de trabajo.

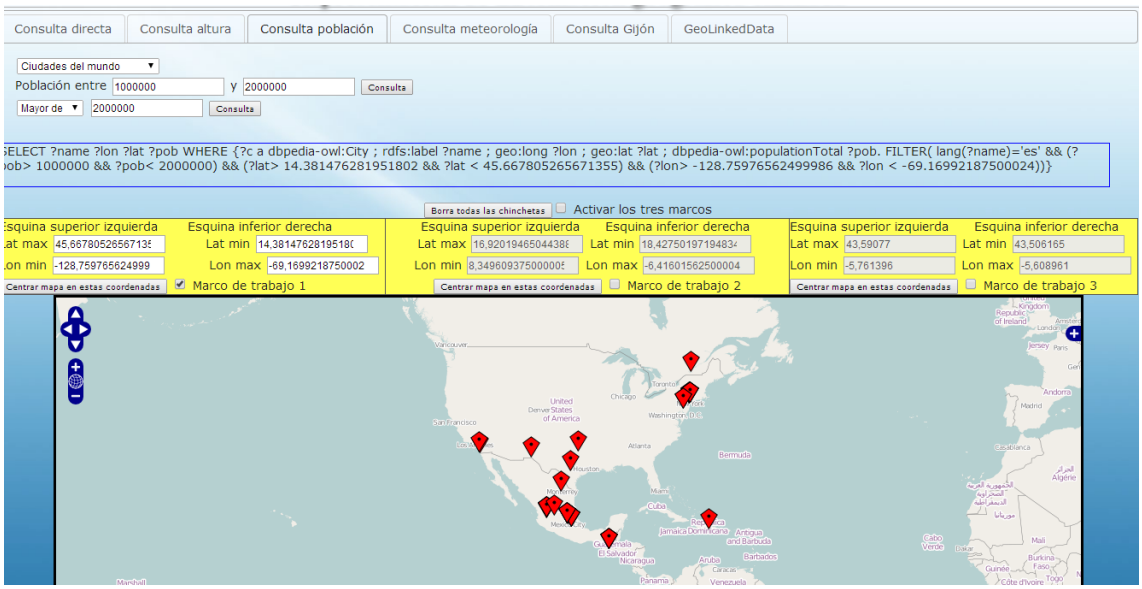
La segunda de las pestañas disponible es “Consulta altura”, y permite consulta la altura de diversos fenómenos geográficos registrados en DBpedia.

En el ejemplo se muestran los volcanes activos de más de 2000 m. de altitud enmarcados en el continente americano:

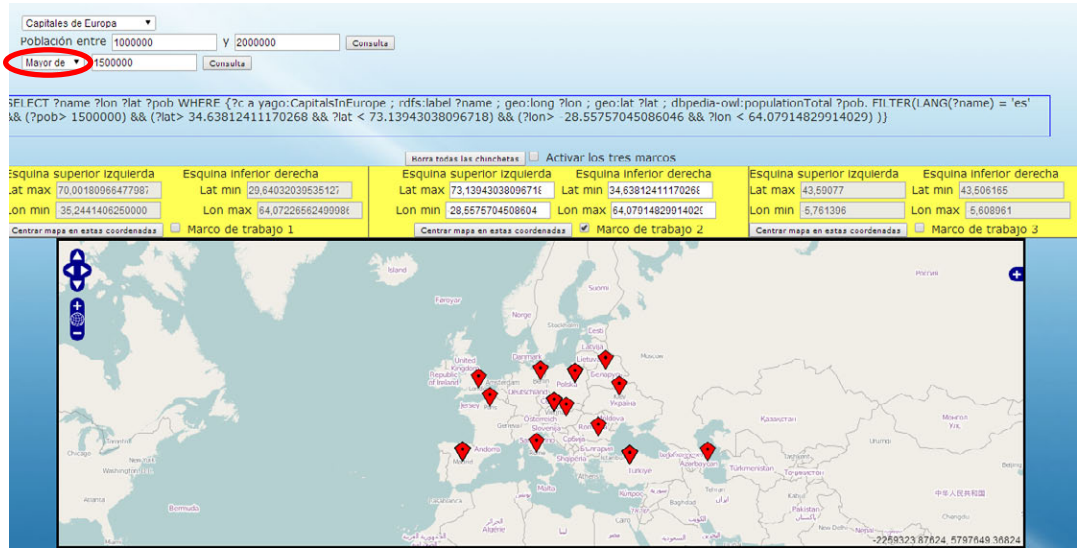


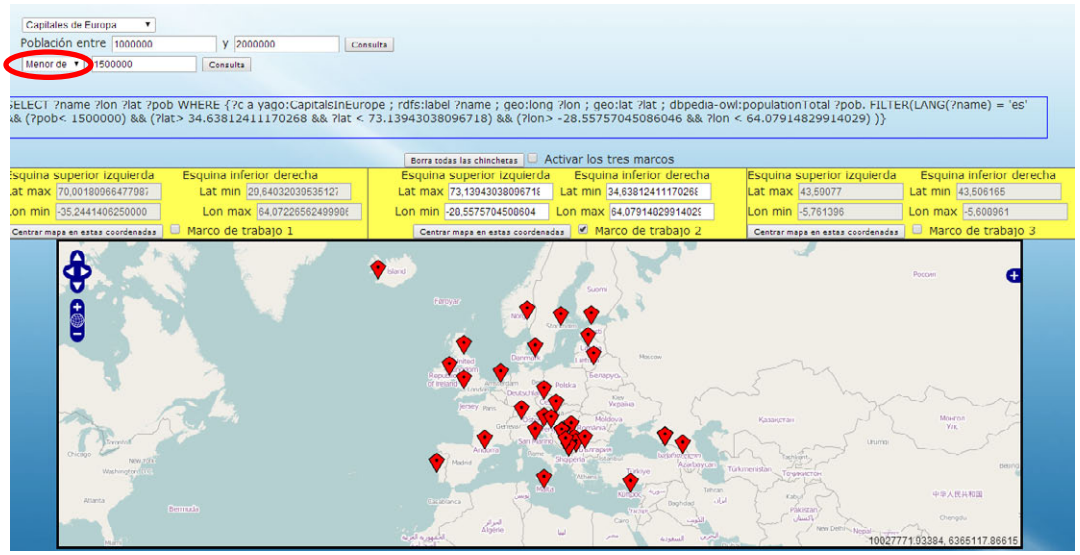
La siguiente de las pestañas es “Consulta población”, que permite consultar ciertas ciudades con alguna característica común y filtrarlas por población. Se puede filtrar por intervalo, o escribiendo una cifra e indicando si se desean representar las ciudades de más o menos habitantes que dicha cifra.

En el ejemplo se muestran las ciudades del mundo etiquetadas como tal por DBpedia, filtradas entre 1.000.000 y 2.000.000 de habitantes en Estados Unidos. El resultado ha sido el siguiente:



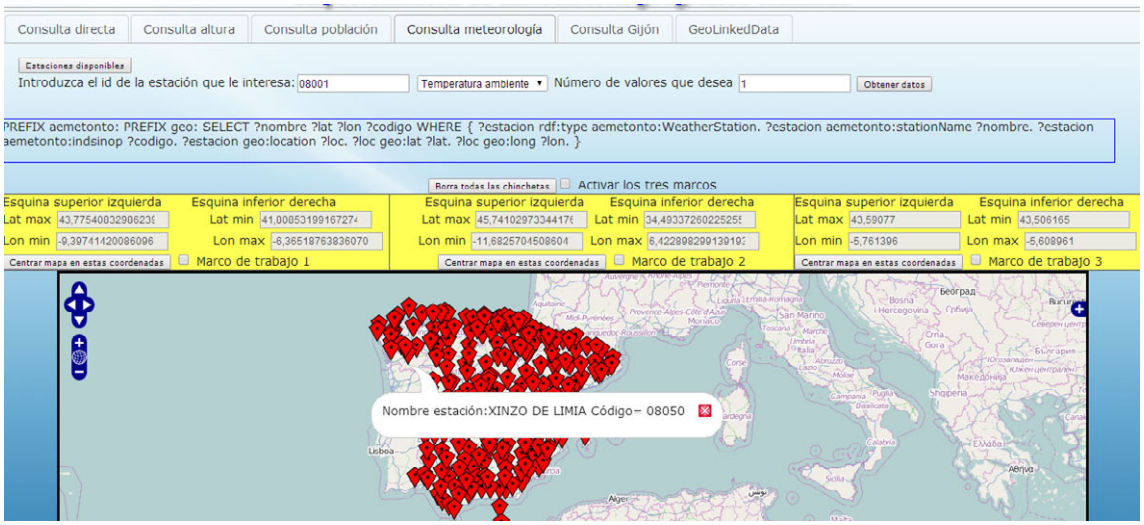
En este otro ejemplo se han seleccionado las capitales europeas de más de 1.500.000 habitantes y las de menos:





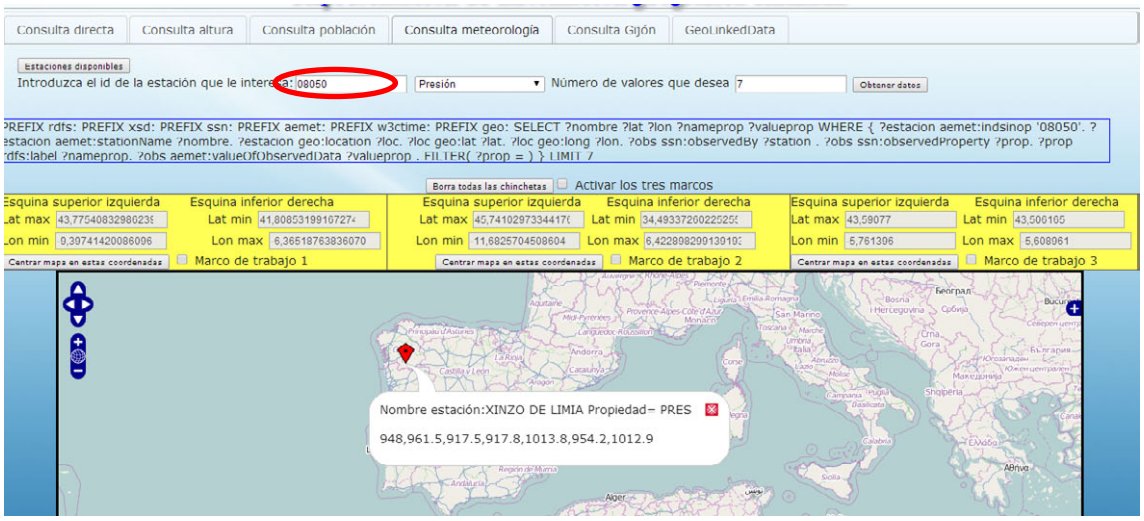
La siguiente de las pestañas permite consultar datos meteorológicos almacenados en el proyecto AEMETLinkedData.

En el caso de este menú se debe comenzar visualizando las estaciones disponibles. Se permite hacerlo en una zona enmarcada en alguno de los marcos de trabajo, o se pueden visualizar todas. Si se desea esto último, simplemente se deben mantener desactivados los marcos. Dentro del PopUp que cada chincheta muestra al hacer clic sobre ella aparece el nombre de la estación y su código.



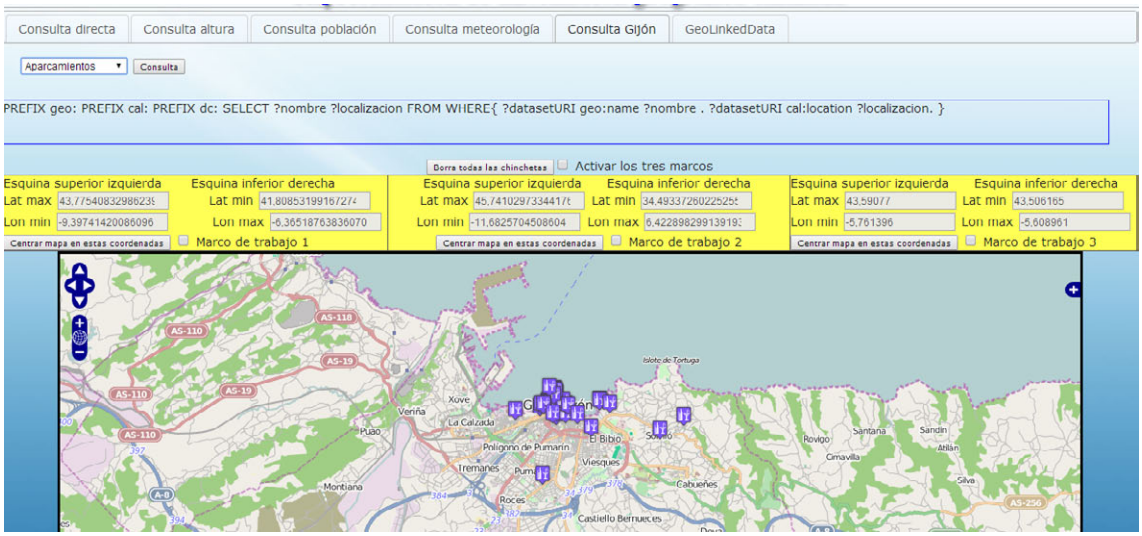
Una vez conocido el código de estación, se puede introducir en la caja de texto marcada en rojo, seleccionar una variable entre las disponibles y el número de valores que se quieren obtener de esa variable en la estación introducida.

Al hacerlo se dibujará tan solo la estación que se haya introducido. En su *PopUp* se podrá visualizar la información de la variable.



En la quinta de las pestañas, se explotan datos de servicios del Ayuntamiento de Gijón. Al ser un ámbito de trabajo relativamente pequeño no tiene validez el módulo de

marcos de trabajo. Simplemente se debe seleccionar el servicio a representar y pulsar el botón "Consulta". A continuación se muestran los aparcamientos de la ciudad de Gijón:



El último de los botones permite representar datos enlazados del Instituto Geográfico Nacional. En el caso de este último ejemplo se representan todos los faros de la costa lucense.

Para ello se selecciona el marco de trabajo, la opción Faro y se pulsa consultar.